2016

Probabilistic methods for quality improvement in high-throughput sequencing data

Xin Yin
Iowa State University

Probabilistic methods for quality improvement in high-throughput sequencing data

by

Xin Yin

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Co-Majors: Bioinformatics and Computational Biology

Statistics

Program of Study Committee:

Karin Dorman, Co-Major Professor

Gregory Phillips, Co-Major Professor

Dan Nettleton

Jarad Niemi

Aditya Ramamoorthy

Iowa State University

Ames, Iowa

2016

DEDICATION

To my grandfathers, Shiqi Yin and Hao Liu.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

The trek to pursue my doctoral degree at Iowa State would not be possible without the helps of many great people, to whom I am genuinely grateful.

First and foremost, my sincere gratitude to my advisors, Dr. Karin S. Dorman and Dr. Gregory Phillips, who have bestowed their profound knowledge, generous support and guidance on me, during my graduate study and research. I would also like to thank Dr. Dan Nettleton, Dr. Jarad Niemi and Dr. Aditya Ramamoorthy, who served as my program of study committee members, for their expertise, insight and help.

Thanks to Zhao Song, Vahid Noroozi and Haijuan Zhang, whom I collaborated with on my thesis projects, for their insightful ideas and contributions. I am also thankful to Dr. Min Wang, Dr. Derek Blythe, Emily King, Luvenia Hellams and Benjamin Mulaosmanovic for their kind help.

Thanks to my colleagues and friends in the BCB program and the department of statistics for their support, encouragement and help over the years.

Lastly, I would like to thank my family for their support and love. Most importantly, I thank my wife, Dr. Cheng Qiu, who has enriched my life, for her unwavering and enlightening support.

ABSTRACT

Advances in high-throughput next-generation sequencing (NGS) technologies have enabled the determination of millions of nucleotide sequences in massive parallelism at affordable costs. Many studies have shown increased error rates over Sanger sequencing, in sequencing data produced by mainstream next-generation sequencing platforms, and have demonstrated the negative impacts of sequencing errors on a wide range of applications of NGS. Thus, it is critically important for primary analysis of sequencing data to produce accurate, high-quality nucleotides for downstream bioinformatics pipelines.

Two bioinformatics problems are dedicated to the direct removal of sequencing errors: base-calling and error-correction. However, existing error correction methods are mostly algorithmic and heuristics. Few methods can address insertion and deletion errors, the dominant error type produced by many platforms. On the other hand, most base-callers do not model the underlying genome structures of the sequencing data, which are necessary for improving base-calling quality especially in low-quality regions. The sequential application of base-caller and error-corrector do not fully offset their shortcomings.

In recognition of these issues, in this dissertation, we propose a probabilistic framework that closely emulate the sequencing-by-synthesis (SBS) process adopted by many NGS platforms. The core idea is to model sequencing data (individual reads, or fluorescent intensities) as independent emissions from a Hidden Markov model (HMM) with transition distributions to model local and double-stranded dependence in the genome, and emission distributions to model the subtle error characteristics of the sequencers. Deriving from this backbone, we develop three novel methods for improving the data quality of high-throughput sequencing: 1) PREMIER, an accurate probabilistic error corrector of substitution errors in Illumina data, 2) PREMIER-bc, an integrated base-caller and error corrector that significantly improves base-calling quality, and

3) PREMIER-indel, an extended error correction method that addresses substitution, insertion and deletion errors for SBS-based sequencers with good empirical performance.

Our foray of using probabilistic methods for base-calling and error correction provides the immediate benefits to downstream analyses with increased sequencing data quality, and more importantly, a flexible and fully-probabilistic basis to go beyond primary analysis.

CHAPTER 1. INTRODUCTION

The cellular activities of all life forms are regulated by genetic sequences encoded in the form of deoxyribonucleic acid (DNA) or ribonucleic acid (RNA). The advent of Sanger sequencing technology in the 1970s marked the milestone achievement to determine the first complete DNA genome, the bacteriophage ϕ-X174 [Sanger et al. (1977)]. The completion of the Human Genome Project in 2004 witnessed the culminating feat of Sanger sequencing. However, the prohibitive cost at the time to sequence the human genome sparked off the development of a wide range of so-called next-generation sequencing (NGS) technologies, that are capable of generating millions of sequences at only a tiny fraction of cost than a decade ago [van Dijk et al. (2014)]. Affordable high-throughput sequencing has contributed to the explosive growth of available sequencing data in public databases [Kodama et al. (2012)] and has revolutionized biological studies with a wide spectrum of applications [Mardis (2008)]. It has not only enabled *de novo* assembly of whole genomes and transcriptomes [Grabherr et al. (2011)], mass sequencing of metagenomes in environmental samples [Shokralla et al. (2012)], and detection of ultra-rare mutations [Schmitt et al. (2012)], but also has supplanted traditional tools in many scientific disciplines. For example, the paradigm shift from microarrays to NGS sequencing empowers base-resolution methylation detection [Laird (2010)], quantification of of transcripts at isoform level [Wang et al. (2009)], and large-scale mapping of DNA-protein interactions [Johnson et al. (2007)].

Challenges arise in analyzing the sequencing data generated by next-generation sequencers. Apart from the drawback of producing sequences with shorter lengths, a major hurdle to overcome is the significantly elevated error rate compared to Sanger sequencing [Quail et al. (2012b)]. The presence of errors in the sequencing data interferes with many downstream analyses, producing biased or noisy results in genome assembly [Salzberg et al. (2012)], variant

calling [Kinde et al. (2011)], transcript quantification [Le et al. (2013)] and taxonomic labeling [Huse et al. (2010)]. It is therefore critically important to develop new bioinformatics tools to control the quality of raw sequencing data in order to ensure the validity of output from biological studies.

This dissertation focuses on the development of statistical methods to denoise high-throughput sequencing data produced by popular next-generation sequencing platforms. We describe interconnected projects that reduce sequencing noise in two essential bioinformatics analyses: base-calling and error correction. A common thread that tightly knits these projects is a Hidden Markov model (HMM), which we utilize to directly model the sequencing process and its error characteristics. We comprehensively study how this HMM can be flexibly adapted to the following problems: (i) correcting substitution errors in Illumina sequencing data; (ii) integrating Illumina base-calling and error correction by directly modeling the raw intensities of Illumina sequencing; and (iii) correcting substitution, insertion and deletion errors, which are found in all major NGS platforms. We demonstrate our probabilistic approach toward NGS quality control with highly competitive performances in all three applications. With the increasingly prominent use of next-generation sequencing technologies in biological and clinical studies, our work not only benefits the downstream studies by producing more accurate sequences, but also provides insights on how to build better and more robust bioinformatics tools for quality control purposes.

Since the subsequent chapters in this dissertation share a lot of recurring topics, we dedicate the remainder of this chapter to review the background knowledge and the work related to our projects.

1.1 Next-generation sequencing

Next-generation sequencing is a broad term, collectively referring to a series of sequencing technologies developed since 2005 [van Dijk et al. (2014)]. Compared with Sanger sequencing, these NGS technologies simplify library preparation by dispensing with vector cloning of the DNA fragments, significantly improve the sequencing throughput by simultaneously monitoring millions of sequencing reactions, and highly automate the determination of nucleotides using

imaging or semiconductor technologies, instead of relying on electrophoresis. Major competitors in the next-generation sequencing market are 454 pyrosequencing by Roche (2005, now discontinued), Illumina/Solexa (2006), SOLiD by Life Technologies (2007, previously Applied Biosystems), Ion Torrent Personal Genome Machine (PGM) (2010, now Life Technologies) and Pacific Biosciences (PacBio) single molecule real-time sequencing (SMRT) (2010). A common denominator of the above sequencing techniques is the reliance on DNA polymerase or ligase. These enzymes are utilized to synthesize deoxynucleotides (dNTPs), which may or may not be labeled, against single-stranded DNA templates. Signals released by base synthesis (hydrogen ions or fluorescent radiation) are "read" by the sequencers and converted into nucleotide sequences (or reads), a common strategy known as sequencing-by-synthesis (SBS).

Imperfections in the biochemistry of nucleotide synthesis, in fluorescent labeling reagents, and in signal measurement and processing may all contribute to the bias, noise and errors in the produced sequencing data.

1.2 Quality control of next-generation sequencing

The success of distilling novel discoveries from any sequencing experiments and subsequent bioinformatics pipelines relies on the high quality of the sequenced nucleotides. To deal with the high error rate of next-generation sequencing in general, both experimental and bioinformatic approaches for quality control are available. On the experimental side, innovative library preparation protocols have been designed to enhance the number of replicates within samples, thus exposing sequencing errors as rare abnormalities. For instance Schmitt et al. (2012) devised a duplex sequencing technique, that attaches unique barcodes to both ends of double-stranded DNA fragments. Subsequent to PCR amplification and DNA sequencing, the barcodes can be used to identify clusters of duplicated reads. Each cluster can then be collapsed into a consensus sequence. Lou et al. (2013) developed circular sequencing, which allows circularized DNA fragments to be repetitively amplified as tandem segments within a read. After sequencing, splitting the reads back into segments creates a pileup of replicated sequences, from which a consensus can be identified. These experimental approaches reduce error rate by multiple

order of magnitudes and allow rare mutations to be identified, but require unconventional library preparations and reduce the effective throughput.

Meanwhile, on the bioinformatics side, there is no shortage of tools dedicated to mitigate or remove sequencing errors from high-throughput sequencing data. A large class of methods employ read filtering and read trimming to discard noisy reads or read segments [Guo et al. (2013), Fabbro et al. (2013)], based on criteria like sequence quality, alignment quality, and variant calling quality. The filtering strategies are broadly matched to distinct flavors of sequencing tasks, such as the sequencing of a transcriptome or metagenome, where options for error removal are scarce. All such filtering methods can suffer from the loss of sequencing coverage due to data removal. Instead of dropping nucleotides with poor quality, an alternative avenue to bioinformatic quality control is through the explicit modeling of sequencing errors. This approach is best exemplified by two categories of primary analysis tools: base-callers that directly model the raw signals of the sequencers and error-correctors that refine the output of the sequencer/base-caller combined duo. A more in-depth review of the two topics follows.

1.2.1 Base-calling

Base-calling software is responsible for converting the raw data produced by sequencers into nucleotides and discretized probabilities measuring the confidence of base-calls, called quality scores. An important function base-callers serve is to characterize noise and errors in the raw signals. Since each sequencing platform exhibits its own distinct error characteristics, base-callers are highly specialized. Throughout this dissertation, our discussion of base-calling centers on the Illumina platform, which currently leads the competitive NGS market.

Illumina sequencers ship with the built-in base-caller, Bustard, which conducts real-time base-calling, turning fluorescent intensities into nucleotides as the sequencing experiment progresses. The benefits of integrated and automated base-calling by Bustard is offset by its simplistic error model. Erlich et al. (2008) were the first to propose a third-party base-caller that improves base-calling accuracy over Bustard, using Support Vector Machines (SVM). Since then, many Illumina base-callers have been developed [Ledergerber and Dessimoz (2011); Cacho et al. (2015)], and they can be separated into two major groups.

Non-model-based methods [Erlich et al. (2008); Kircher et al. (2009); Renaud et al. (2013)] elect to use machine learning techniques to predict the nucleotides, using features extracted from the intensity data. These methods require control samples with known reference to be sequenced either as spike-in or on a different tile. The mapped sequences (with errors) and intensities are used as a training set to train SVM or logistic regression classifiers, which in turn generate predictions of nucleotides from the intensity data of interest.

Model-based base-callers [Rougemont et al. (2008); Kao et al. (2009); Bravo and Irizarry (2010); Massingham and Goldman (2012); Das and Vikalo (2013)] explicitly model the common error characteristics of the Illumina sequencing process. For a given read of length l, these models can all be considered special cases of the following general form [Cacho et al. (2015)]:

$$\boldsymbol{Y}_i = \boldsymbol{C}\boldsymbol{X}_i\boldsymbol{P}\boldsymbol{D} + \boldsymbol{E}_i, \tag{1.1}$$

where $\underset{l\times 4}{\boldsymbol{Y}_i}$ is a matrix of observed fluorescent intensities in four channels representing A, C, G and T labels, $\underset{l\times 4}{\boldsymbol{X}_i}$ is the latent intensity values or nucleotide indicators, $\underset{4\times 4}{\boldsymbol{C}}$, $\underset{l\times l}{\boldsymbol{P}}$ and $\underset{l\times l}{\boldsymbol{D}}$ respectively model the cross-talk (overlap of fluorescent spectra in different channels), phasing/prephasing effects (base synthesis lagging or leading base determination) and signal decay (DNA degradation and attrition over time), and $\underset{l\times 4}{\boldsymbol{E}_i}$ is the error term. The goal of base-calling, is to properly estimate the free parameters in matrices $\boldsymbol{C}, \boldsymbol{P}$, $\boldsymbol{D}$ and $\boldsymbol{E}_i$, and find $\boldsymbol{X}_i$ that best explains the observed signals $\boldsymbol{Y}_i$.

All above methods, except Bustard, are considered "off-line" base-callers, since the estimation and base-calling are conducted after the sequencing process completes. Interestingly, all these base-callers focused on only modeling $\boldsymbol{Y}_i$, while ignoring any structure in the latent sequence $\boldsymbol{X}_i$. Since the genome is read with high coverage, the set, $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$ comes from a strictly finite set and is highly replicated, a fact that provides valuable information when signals in $\boldsymbol{Y}_i$ are of poor quality.

To reduce the size of base-calling output, base-callers often condense the posterior probabilities of $\boldsymbol{X}_i \mid \boldsymbol{Y}_i$ into single nucleotides, *i.e.* the column index that maximizes individual row vectors. The uncertainty of each base-call is also converted from a probability into discrete

quality scores. As a result, base-callers produce sequences in standard FASTQ formats, in which both nucleotides and quality scores can be represented by single ASCII characters.

1.2.2 Error correction

Unlike base-calling, which must be tailored to the source sequencing platform, error correction operates by intercepting the output of base-callers, which are sequences of nucleotides and quality scores (called reads) in standard FASTQ format. The decoupling of error correction from sequencing platforms plus the ubiquitousness of the FASTQ format, contribute the popularity of error correction methods. Recent surveys of this field [Yang et al. (2013), Molnar and Ilie (2014), Laehnemann et al. (2015), Alic et al. (2016)] covered a diverse range of error correction methods.

The prospect of error correction is founded on the premise that each nucleotide in the sequenced genome is expected to be repetitively covered by multiple short reads. Reconstructing a multiple sequence alignment of reads reveals random errors as low-occurrence outliers deviating from the position-wise consensus [)]. The idea is easy in principle, but computationally challenging since the positional information of reads is unknown, unless a reference sequence that facilitates the alignment is known *a priori*. In fact, since most sequencing experiments do not resequence a perfectly known genome, as interest often lies in sequencing genomes *de novo* or discovering new variants, most error correction applications are either entirely reference-free or directionally biased by the use of a reference.

Although there is great redundancy in NGS data, a key challenge to reference-free error correction is that sequence redundancy is lacking at read-level, because reads are derived from random fragments of the genome, and throughput is not so great to replicate the same fragment often. Instead, consider substrings of reads of length k, which are called $k$mers. When k is sufficiently smaller than the read length, adequate redundancy for error correction may be reattained at the $k$mer-level. Stemming from this basic idea, the strategies adopted by existing reference-free error correction methods can be divided into three major categories.

Multiple-sequence alignment (MSA) based methods attempt to construct approximate alignment of reads by grouping reads that share at least one common $k$mer. Within each group of

presumably overlapping reads, Coral [Salmela and Schröder (2011)] iteratively aligns each read to the consensus sequence, which is initialized by the first read, using a modified Needleman-Wunsch algorithm with no end-gap penalties. The consensus is updated after each alignment, and is eventually used for identifying errors within group. ECHO [Kao et al. (2011)] computes the consensus within each group using an expectation-maximization (algorithm). DAGCon [Chin et al. (2013)] and Karect [Allam et al. (2015)] construct a directed acyclic graph (DAG) from the overlapped reads, and identify the consensus as the mostly likely path through the DAG.

The second category is k-spectrum based methods. Instead of forming read alignments, these methods rely on the assumption that when k is adequately large, all observed $k$mers in the reads (known as the k-spectrum) are each increasingly likely to derive from a unique location in the genome. Since the genome size is finite and can only supply limited number of *true* $k$mers, the true k-spectrum should be highly sparse, with large distances between $k$mers. Sequencing errors mutate the valid $k$mers, populating erroneous $k$mers into the k-spectrum, which appear as tight "clouds" surrounding true $k$mers. By computing the observed frequency of $k$mers as sufficient statistics, $k$mers with rare occurrence are considered erroneous, and can be corrected to neighboring high-frequency $k$mers. This idea was seen in many earlier methods, like Reptile [Yang et al. (2010)], Quake [Kelley et al. (2010)] and Hammer [Medvedev et al. (2011)]. If the genome is highly repetitive, the same $k$mer may appear at multiple genomic loci. To resolve this ambiguity, context information is often used in more recent methods, like Musket [Liu et al. (2013)], BLESS [Heo et al. (2014)] and Blue [Greenfield et al. (2014)], by implicitly considering paths of $k$mers achieves high-multiplicities in the de Bruijn graph [Pevzner et al. (2001) as candidate corrections for a read segment with low-frequency. To further reduce computational complexity, Liu et al. (2013) and Heo et al. (2014) employed Bloom filter [Bloom (1970)] to aggressively filter out low-multiplicity $k$mers from the k-spectrum.

Finally, suffix-tree/array based methods, like SHREC [Schröder et al. (2009)], HSHREC [Salmela (2010)] and HiTEC [Ilie et al. (2011)], can be considered as generalized k-spectrum methods, however with variable k. By constructing a suffix-tree/array from the read sequences, sequencing errors are detected as tree nodes or branches with low weights, and can be corrected

to sibling branches with abundant weights. These methods usually have a heavy memory footprint to store the suffix tree/array, and suffer from the same ambiguity issues when the $k$mer uniqueness assumption does not hold. A more recent method, Fiona [Schulz et al. (2014)] alleviated both problems by utilizing partial suffix trees, and alignments between erroneous subtrees and correct subtrees to find the optimal correction.

Another major differentiator of error correction methods is the type of errors that can be corrected. Illumina is the only NGS platform that predominantly produces substitutive errors. Partly attributed to the dominance of Illumina sequencers in the market, the popularity of substitution error correction methods is also due to the ease of modeling. Most k-spectrum and suffix-tree/array based methods can only handle substitution errors, with the exceptions of Blue and Fiona. If a $k$mer contains n substitution errors, the plausible candidates to correct it can be localized in the k-spectrum efficiently by computing the Hamming distance between the target and the candidate. Single-instruction, multiple data (SIMD) instructions in modern processors [Calonder et al. (2012)], and efficient algorithms [Mann (2015)] exist to accelerate the computation of Hamming distance and efficient error correction algorithms can compute the distance recursively using upstream contexts as well [Liu et al. (2013), Heo et al. (2014)]. Insertion and deletion errors (or indels), on the other hand, introduce shifts in the erroneous sequence, and must be evaluated with the edit distance [Navarro (2001)]. Searching for candidate $k$mers in the edit distance space is computational extensive, and is only tractable with some heuristics, like depth-first searches [Greenfield et al. (2014)]. Schulz et al. (2014) utilize alignments between subtrees to correct indels, but only within limited edit distance. For this reason, MSA-based methods are generally better poised to handle insertion and deletion errors. Despite the computational challenge, correcting indel errors is critically important. Indel errors occur in all platforms, albeit less frequently in Illumina data, and are the dominant errors for platforms like 454, Ion Torrent and PacBio [Quail et al. (2012b); Alic et al. (2016)]. In addition, indel errors can confound the discovery of true indel variants in the genome [Bragg et al. (2013)], which are found to be functionally important [Mullaney et al. (2010)].

1.3 Denoising of NGS data using Hidden Markov model

Notwithstanding the many advances of base calling and error correction algorithms, there is ample room for improvements in this field. Most error correction software are algorithmic and heuristic, assuming primitive error models (*e.g.* uniform error rate), and often ignore the quality score information passed by the upstream base-caller. Correspondingly, the majority of base-callers fail to utilize genome structures, which facilitate with the accurate determination of bases, particularly in the 3' ends of reads where intensities become noisy. Such obvious bias in modeling choices pertaining to either application is often delibrate to encourage modular design of bioinformatics tools, however may lead to undesirable consequences. With informative data left unutilized, this sequential analysis of sequencing data is prone to leave uncorrected errors in the gap between the two stages.

After extensive study of the existing base-calling and error correction methods, we proposed a probabilistic framework that closely resembles the sequencing-by-synthesis principles of mainstream NGS sequencers. Under the umbrella of this framework, we present three different projects in subsequent chapters, before concluding this dissertation with discussions of our probabilitic approach, and future directions.

In chapter 2, we present PREMIER, short for PRobabilistic Error-correction using Markov Inference in Errored Reads. We motivate a Hidden Markov modeling approach toward error correction of substitution errors in Illumina sequencing data, by modeling the reads produced by Illumina sequencers as independent emissions from the generative model. We describe our modeling choices to account for the non-randomness of genomic sequence, the dependence between the two antiparallel DNA strands, and the bias in substitutive errors. Most critically, we propose a regularized estimation procedure to address the overparameterization of our model caused by sequencing errors. We cover the numerical algorithms to solve the optimization of the penalized log-likelihood of the HMM. We extensively study the effects of the model parameters, including the order of the Markov chain, the penalty parameters, and model complexity parameters, and then empirically establish procedures to choose model parameters based on the traits of the input dataset. Finally, we compare the error correction performance of PREMIER

against state-of-the-art error correction methods, and demonstrate competitive and robust results. PREMIER serves as a base framework to this dissertation from two perspectives. On the modeling side, the components of the HMM are easily extended, as we demonstrate in later chapters, to solve more complex problems. On the computational side, our implementation of PREMIER was parallelized using OpenMP, and was highly optimized for memory usage and computational efficiency, easing development of subsequent projects.

In chapter 3, inspired by the usage of HMM for Illumina base-calling [Das and Vikalo (2013)], we develop extensions of the emission distribution of the HMM, turning our model into a unified base-caller and error corrector. The modified emission distribution shares the backbone with many other model-based Illumina base-callers to model the common sequencing error properties of Illumina sequencer: cross-talk, signal decay and residual effects. We discuss our simplifications to the base-calling component of our model, which allow us to efficiently estimate model parameters using dynamic programming algorithms and the alternating expectation conditional-maximization (AECM) framework [Meng and Van Dyk (1997)]. We then highlight the improvements in base-calling quality attained by our software, PREMIER-bc, over not only all existing Illumina base-callers, but also the sequential application of base-calling plus error correction.

Lastly, in chapter 4, we address the error correction of insertion and deletion errors, by developing new software, PREMIER-indel. We introduce special hidden states to the state space of the PREMIER HMM to model the desynchronization events between base synthesis and base calling. We then reparameterize the transition probabilities as mixture distributions of genomic transition probabilities and insertion/deletion error parameters. This allows us to reuse the penalized estimation scheme in PREMIER to estimate the genomic transition probabilities, without adding many new parameters to the model. A modified Viterbi algorithm was developed to specifically address the long bursts of insertion errors found in Ion Torrent sequencing datasets. Following the discussion of model derivation and implementation details, we show that PREMIER-indel consistently outperforms competing indel error correctors.

CHAPTER 2. PREMIER: PROBABILISTIC ERROR-CORRECTION USING MARKOV INFERENCE IN ERRORED READS

Abstract

The presence of errors in next-generation sequencing (NGS) data raises challenges for downstream analyses and has motivated the development of multiple error correction algorithms that improve data quality and downstream results. Many existing methods share the same core idea of utilizing highly duplicated short strings up to length k (or $k$mers) to identify errors within short locality, and can be viewed as heuristic approximations to a Hidden Markov Model (HMM), which closely resembles the true generative process of mainstream NGS platforms that operate on sequencing-by-synthesis principles. In this work, we develop PREMIER, a flexible probabilistic method for error correction in Illumina data. By modeling the short reads produced by the sequencer as independent realizations of an HMM, our model is able to capture the local and double-stranded dependence inherent in DNA sequences, while accommodating nuances in the error characteristics of the sequencer. Using real Illumina datasets, our analysis show that PREMIER significantly improves on the top error correction methods founded on the k-local discipline, and is highly competitive with alignment-based methods that can use contexts beyond $k$mers.

2.1 Introduction

The development of next-generation sequencing (NGS) technology has revolutionized the biological sciences by making genomic data affordable and efficient to sequence [van Dijk et al. (2014)]. The low cost and high throughput of these sequencing platforms is offset, however, by an elevated error rate relative to older Sanger sequencing technology [Glenn (2011)]. Error

contaminated sequence reads complicate many downstream analyses, including detection of ultra-rare mutations [Schmitt et al. (2012)], genetic heterogeneity detection [Lou et al. (2013)] and *de novo* assembly [Salzberg et al. (2012)].

Errors are introduced at multiple points during the sequencing process [Schmitt et al. (2012)]. Some errors, such as misincorporations during an early round of PCR amplification are difficult to detect [Schmitt et al. (2012)], but most errors introduce detectable patterns that can be used to winnow errors in data post-processing. In fact, a plethora of error correction methods take advantage of these patterns to detect and correct sequencing errors [Molnar and Ilie (2014)]. These methods intercept reads after base calling, a convenient moment to intercalate in the NGS pipeline since sequencing data share a universal format across platforms after base calling and occupy much less space than upstream data.

Each base-called read is a best estimate of a contiguous portion of a genome, from a few dozen up to several thousand base pairs long. The starting position of each read is random. By repeatedly sampling the finite genome thousands to billions of times, each genomic position is covered by multiple reads, engendering redundancy that is key to error correction. In principle, alignment of the reads can identify genomic signal and separate it from sporadic errors, but without the genomic position of each read, the sheer size of the data makes this task infeasible in the absence of a reference genome.

Assembly reconstructs genomes from fragment reads by using local alignment information. For short reads, the dominant method uses de Bruijn graphs [Pevzner et al. (2001)], where vertices represent nucleotide strings of length k (called $k$mers) and edges connect vertices with a $k-1$ matching suffix/prefix and the unobserved genome is a path through the graph. The de Bruijn graph constructed from a read set is sparse because only observed $k$mers and edges are included. Because of errors in the reads (some $k$mers may not exist in the genome) and repetition in the genome (not all $k$mers are unique), some vertices will have multiple incoming and outgoing edges. Disentangling these knots in the graph is the task of genome assembly/error correction.

To correct errors in a probabilistic framework, we propose to model the sequencing process as a Hidden Markov Model on a sparse de Bruijn graph (HMMSdBG). Thus, each read is

independently and identically generated from a genomic de Bruijn graph with error. The generative process randomly picks a starting vertex ($k$mer) in proportion to its abundance in the genome and traverses edges in a probabilistic fashion. At each vertex, the model outputs a noisy base call and a quality score, informing on the call quality. The HMMSdBG and the true generative process are remarkably well-matched. Both the sequencer and most base callers move along a genome fragment, "observing" only limited context information.

This article describes PREMIER, a flexible, fully-probabilistic method of error correction for Illumina sequencing data using the proposed HMMSdBG framework. PREMIER estimates the parameters of the HMMSdBG from the observed reads, then corrects each read to its maximum-likelihood state sequence. Our method removes more errors and introduces fewer new errors than all existing $k$mer-based error correction methods. There are several modeling choices and techniques key to the success of our method. In particular, we simultaneously use information from both forward and reverse sequences, halving the number of model parameters, and we regularize the transition probabilities to reflect $k$mer uniqueness and genome finiteness.

Many existing error correction methods are heuristic approximations to the HMMSdBG. Among top performers in a recent comparison [Molnar and Ilie (2014)], BLESS [Heo et al. (2014)], Musket [Liu et al. (2013)], and HiTEC [Ilie et al. (2011)] identify the most likely trajectory for a read based on nearby heavily traversed pathways. All these methods are $k$mer-based methods. Alignment-based methods, such as Coral [Salmela and Schröder (2011)] and Karect [Allam et al. (2015)], make use of long distance information up to the end of the read and can theoretically achieve better performance than local $k$mer-based methods, although they too use $k$mers to seed their alignments.

PREMIER, implemented in C/C++, comfortably beats all $k$mer-based methods and is competitive with Karect. As a flexible, probability model of sequencing technology, it has the potential to be more than an error correction method. It not only adapts easily to new sequencing kits and technologies, but it also has the potential improve base calling, genotype calling, variant calling, genome and transcript abundance quantification, rare mutant and alternative spliced transcript detection, use context beyond the $k$mer, like Karect, and more.

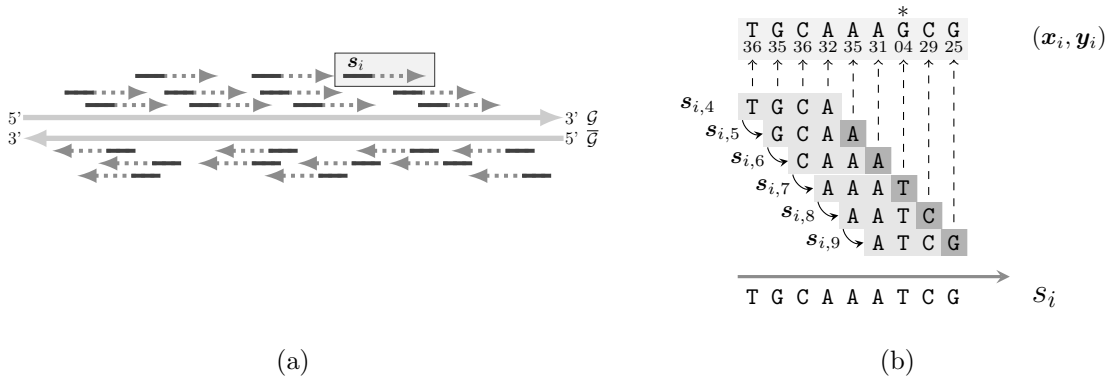
2.2 Methods

2.2.1 Hidden Markov Modeling of Error Correction

The forward strand of a target genome denoted $\mathcal{G}$, is a quaternary sequence of length $|\mathcal{G}|$ over the alphabet $\Omega = \{A, C, G, T\}$. The reverse complement of $\mathcal{G}$, $\overline{\mathcal{G}}$, consisting of complementary bases ($A \Leftrightarrow T, C \Leftrightarrow G$) in reverse order, yields the reverse strand. The sequencer produces reads from short fragments of the genome, either moving along strand $\mathcal{G}$ or in the reverse direction along strand $\overline{\mathcal{G}}$ (Fig. 2.1(a)). Thus, the i-th read is an estimate of a substring $\mathbf{s}_i$ that starts at a random position in either $\mathcal{G}$ or $\overline{\mathcal{G}}$, although the strand-type cannot be known. The output of the i-th read is the two-tuple $(\mathbf{x}_i, \mathbf{y}_i)$ of nucleotides $\mathbf{x}_i$ and quality scores $\mathbf{y}_i$, both of length l. The elements of $\mathbf{x}_i$ are "base calls" in Ω. The elements of $\mathbf{y}_i$ are "quality scores", i.e., discrete, per-base reliability assessments of the corresponding base call. The entire set of sequence reads and quality scores produced by a sequencing experiment is denoted by $\mathcal{R}$.

The main idea in PREMIER is to model each observed read $(\mathbf{x}_i, \mathbf{y}_i)$ as an independent emission from a Hidden Markov Model (HMM) where the states are $k$mers, strings of length k on alphabet Ω, arranged in a sparse de Bruijn graph. The emission distribution models the error properties of the upstream data analysis pipeline, including library preparation, the sequencer, and base calling, and can account for properties such as increasing error rates with read length [Minoche et al. (2011)], and the distribution of quality scores. The Markov chain models $\mathcal{G}$.

A Markov chain along a sparse de Bruijn graph is not irreducible; many transitions are unique by virtue of the finiteness of the genome and large k. Information about model parameters comes from the fact that the different positions in the genome are sequenced multiple times with high probability. The critical sample size which determines the success of our method is the number of times that each nucleotide is observed. Under the assumption of uniformly distributed starting points for the reads (typically referred to as the uniform coverage assumption), this quantity is called the coverage, $c = \frac{|\mathcal{R}|l}{|\mathcal{G}|}$. With sufficient coverage to fit the model parameters accurately, we can determine the maximum likelihood state sequence $\widehat{\mathbf{s}}_i$ that best explains $(\mathbf{x}_i, \mathbf{y}_i)$, and we declare it the *corrected* read.



(a) (b)

Figure 2.2: A schematic of the Hidden Markov Model for error correction.

(a) Multiple copies of a single genome, which consists of two complementary and antiparallel strands, are broken into short fragments (dashed arrows). The sequencer reads the portion labeled $\mathbf{s}_i$, of each fragment (solid segment) in the $5'$ to $3'$ direction, producing the observable read $(\mathbf{x}_i, \mathbf{y}_i)$. The production of a single read (highlighted box) is modeled in (b) as the realization of an HMM. The hidden state sequence $\mathbf{s}_i = \texttt{TGCAAATCG}$ is decomposed into overlapping $k$mers, here shown with $k = 4$. The sequencer starts in state `TGCA` and emits bases and quality scores (shown at top) for all four positions. Then, it advances one nucleotide to the next $k$mer and emits one base and quality score for the last position in the $k$mer. After finishing, it produces a read $(\mathbf{x}_i, \mathbf{y}_i)$ with one error, a `G` substituted for `T` at position seven. The low quality score at position seven indicates a problem, which is corroborated if $k$mer `AAAG` is rarely observed. Specifically, if transition `CAAA`→`G` is observed much less frequently than `CAAA`→`T`, then `G` can be corrected to `T`.

Let $\mathcal{S}$ be the set of true sequences that generates the reads $\mathcal{R}$. Given $\mathbf{s}_i \in \mathcal{S}$ that generates data $(\mathbf{x}_i, \mathbf{y}_i)$, we let $\mathbf{s}_{i[j]}$ denote the j-th nucleotide of $\mathbf{s}_i$ and $\mathbf{s}_{i[j...m]}$ denote the substring of $\mathbf{s}_i$ from position j to position m (both positions j and m are included). Substrings of $\mathbf{s}_i$ with length k are called $k$mers. The $k$mer ending at position t, is denoted by $\mathbf{s}_{i,t} \equiv \mathbf{s}_{i[t-k+1...t]}$. It follows that we can decompose $\mathbf{s}_i$ into $l - k + 1$ overlapping $k$mers, $\mathbf{s}_{i,k}, \mathbf{s}_{i,k+1}, \ldots, \mathbf{s}_{i,l}$, where any two adjacent $k$mers $\mathbf{s}_{i,t}$ and $\mathbf{s}_{i,t+1}$ share a common $k - 1$ nucleotide suffix/prefix. These $k$mers are the hidden *states* of the HMM. We can similarly decompose the observed reads ($\mathbf{x}_i$ into $\mathbf{x}_{i,k}, \mathbf{x}_{i,k+1}, \ldots, \mathbf{x}_{i,l}$) and quality scores ($\mathbf{y}_i$ into $\mathbf{y}_{i,k}, \mathbf{y}_{i,k+1}, \ldots, \mathbf{y}_{i,l}$).

We assume that the underlying Markov process starts at state $\mathbf{s}_{i,k}$ with probability governed by an initial state distribution. Given the initial state, the sequencer emits k base calls and quality scores, $(\mathbf{x}_{i,k}, \mathbf{y}_{i,k})$. The subsequent sequencing cycles are indexed by discrete positions $t = k + 1, k + 2, \ldots, l$. At the t-th cycle, the sequencer transitions from state $\mathbf{s}_{i,t-1}$ to state

$\mathbf{s}_{i,t}$ and emits $(\mathbf{x}_i[t], \mathbf{y}_i[t])$, reflecting the sequencer's best guess of $\mathbf{s}_i[t]$ and its confidence (see Fig. 2.1(b)).

There are critical modeling choices that make our approach work, and all revolve around the central notion of a $k$mer. The genome $\mathcal{G}$ is finite, and each $k$mer occurs 0 or some finite number of times in $\mathcal{G}$. If k is small, the set of genomic $k$mers is dense, most $k$mers occur in multiple genomic locations, and transition probabilities are not useful for separating genomic variation from error. As k increases, the set of genomic $k$mers becomes sparse, most $k$mers become unique in the genome, and transition probabilities become highly informative. The k required to guarantee unique $k$mers is typically longer than the read length l or yields $k$mers with insufficient coverage to distinguish error and true transitions. The following three ideas yield a compromise solution with excellent performance. (1) We only include in the state space those $k$mers that have been observed in the reads (and their reverse complements). (2) The resulting state space includes many error $k$mers, and the resulting model is overparameterized, so we impose a ℓ_0-like penalty on the transition parameters to enforce our belief that most error-free $k$mers come from unique locations in the genome. (3) Finally, we effectively double coverage by utilizing information from both the forward and reverse complement strands of the genome. This model, and many others, would work perfectly under the ideal scenario of unique $k$mers with high and uniform coverage, well above the error rate. Our method has the extra advantage that it can use weaker signals present in the read set $\mathcal{R}$ to make accurate decisions when these assumptions are violated. We now describe the components of the HMM in detail.

2.2.1.1 State space

Define the observed k-spectrum, $\mathcal{K}^O$, as the set of all observed $k$mers in the read set $\mathcal{R}$, *i.e.*

$$\mathcal{K}^O = \bigcup_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{R}} \{\mathbf{x}_{i,k}, \mathbf{x}_{i,k+1}, \ldots, \mathbf{x}_{i,l}\}.$$

We assume the state space of the HMM is

$$\mathcal{K} = \mathcal{K}^O \cup \{\overline{\boldsymbol{\omega}} : \forall \boldsymbol{\omega} \in \mathcal{K}^O\}, \tag{2.1}$$

where $\overline{\boldsymbol{\omega}}$ is the reverse complement of $\boldsymbol{\omega}$ and is deliberately inserted into $\mathcal{K}$ to reduce the risk of excluding true $k$mers, a likely scenario in low coverage datasets. The state space size is

capped well below the worst case, $|\mathcal{K}| \ll 4^k$, for large enough k, substantially decreasing the complexity of the HMM.

2.2.1.2 Transition distribution

We model the hidden states as a k-th order Markov chain. The transition matrix can be parameterized as a $|\mathcal{K}| \times |\mathcal{K}|$ matrix with elements $p(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ for $\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{K}$. On occasion, we will write $p(\boldsymbol{\nu}_{[k]} \mid \boldsymbol{\omega})$ or simply $p(b \mid \boldsymbol{\omega})$ for $b \in \Omega$ since the first $k-1$ bases in $\boldsymbol{\nu}$ necessarily match $\boldsymbol{\omega}_{[2\ldots k]}$. Assuming that any true transition in the genome will have been observed at least once without error, we constrain the transition probabilities by setting $p(\boldsymbol{\nu} \mid \boldsymbol{\omega}) \equiv 0$ if the transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ is never observed in the read set. If $\mathcal{T}(\boldsymbol{\omega})$ is the set of $\boldsymbol{\nu} \in \mathcal{K}$ such that $\boldsymbol{\omega} \to \boldsymbol{\nu}$ or $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$ is observed at least once in $\mathcal{R}$, then the transition probabilities satisfy constraint

$$\sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} p(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = 1, \quad \forall\, \boldsymbol{\omega} \in \mathcal{K}. \tag{2.2}$$

This transition matrix is sparse, with at most four non-zero transitions per row and usually only one, for adequately large k.

The pair $\widetilde{\boldsymbol{\omega}} = (\boldsymbol{\omega}, \overline{\boldsymbol{\omega}})$ is called a *canonical kmer* when ordered lexically [Liu et al. (2013)]. If coverage is uniform across both strands (and ignoring minor genome end effects), then a read starts in state $\boldsymbol{\omega}$ or $\overline{\boldsymbol{\omega}}$ with equal probability. We denote this initial state probability for $\widetilde{\boldsymbol{\omega}}$ as $\mu(\widetilde{\boldsymbol{\omega}})$. Note that if transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ occurs uniquely on the forward strand, then $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$ occurs uniquely on the reverse strand, and uniform coverage implies

$$\mu(\widetilde{\boldsymbol{\omega}}) \cdot p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = \mu(\widetilde{\boldsymbol{\nu}}) \cdot p_2(\overline{\boldsymbol{\omega}} \mid \overline{\boldsymbol{\nu}}), \tag{2.3}$$

where we label the transitions as *1* or *2* according to their strand. If we knew the strand of each transition, our model would account for the complementary nature of the DNA strands. In particular, compared to a strand-naive model that has independent transition and initial state distribution parameters for the forward and reverse strands, our model would require only half as many parameters.

However, not all $k$mers are unique and even when they are, the strand to which a $k$mer belongs is unknown to us. Fortunately, Eq. (2.3) still holds when transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ is not

unique, and even if it appears on both forward and reverse strands. Theoretically, all we need is a consistent method to label transitions such that when transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ is labeled *1*, transition $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$ is labeled *2* and $p_2\left(\overline{\boldsymbol{\omega}} \mid \overline{\boldsymbol{\nu}}\right)$ is defined in terms of $p_1\left(\boldsymbol{\nu} \mid \boldsymbol{\omega}\right)$.

2.2.1.3 Emission distribution

We use the emission distribution to model the error properties and quality scores of the sequencer and to reduce computational complexity by limiting the number of plausible hidden state sequences. Sequencing errors are implied if $\mathbf{x}_i[j] \neq \mathbf{s}_i[j]$, for any $1 \leq j \leq l$. The number of errors in a $k$mer $\mathbf{x}_{i,t}$ is given by the Hamming distance function $D(\mathbf{s}_{i,t}, \mathbf{x}_{i,t})$. We assume that errors are rare and impose constraints on the maximum number of allowed errors in any $k$mer. State sequences $\mathbf{s}_i$ that cannot emit $\mathbf{x}_i$ are eliminated from consideration.

Let $\mathcal{N}_{it}(\mathbf{x}_{i,t})$ (the neighborhood of $\mathbf{x}_{i,t}$) be the collection of $k$mers $\boldsymbol{\omega} \in \mathcal{K}$ that could have emitted observed state $\mathbf{x}_{i,t}$,

$$\mathcal{N}_{it}(\mathbf{x}_{i,t}) = \{\boldsymbol{\nu} : \boldsymbol{\nu} \in \mathcal{K} \text{ and } D(\mathbf{x}_{i,t}, \boldsymbol{\nu}) \leq d_{it}\}. \tag{2.4}$$

Neighborhood size is limited by the maximum Hamming distance d_{it}, a parameter that depends on both the read and the read position. Our algorithm for choosing d_{it} balances between computational overhead and error-correction capabilities and is discussed in the Supplementary Information (§A.2.2.)

For the first $k$mer, we assume that the elements of $(\mathbf{x}_{i,k}, \mathbf{y}_{i,k})$ are emitted according to

$$f_k(\mathbf{x}_{i,k}, \mathbf{y}_{i,k} \mid \mathbf{s}_{i,k}) \propto \mathbb{1}\{\mathbf{x}_{i,k} \in \mathcal{N}_{ik}(\mathbf{s}_{i,k})\} \times$$
$$\prod_{j=1}^{k} \left(g_j(\mathbf{x}_i[j] \mid \mathbf{s}_i[j]) \times \begin{cases} q_{ej}\left(\mathbf{y}_i[j]\right) & \text{if } \mathbf{x}_i[j] \neq \mathbf{s}_i[j] \\ q_{cj}\left(\mathbf{y}_i[j]\right) & \text{if } \mathbf{x}_i[j] = \mathbf{s}_i[j] \end{cases} \right).$$

For subsequent $k$mers, $\mathbf{s}_{i,t}$ only emits $(\mathbf{x}_i[t], \mathbf{y}_i[t])$, since the $k-1$ prefixes $\mathbf{x}_{i,t}^- \equiv \mathbf{x}_{i,t[1\ldots k-1]}, \mathbf{y}_{i,t}^- \equiv \mathbf{y}_{i,t[1\ldots k-1]}$ have already been emitted. Therefore, for $k < t \leq l$,

$$f_t(\mathbf{x}_i[t], \mathbf{y}_i[t] \mid \mathbf{x}_{i,t}^-, \mathbf{s}_{i,t}) \propto \mathbb{1}\{\mathbf{x}_{i,t} \in \mathcal{N}_{it}(\mathbf{s}_{i,t})\} \times$$
$$g_t(\mathbf{x}_i[t] \mid \mathbf{s}_i[t]) \times \begin{cases} q_{ct}\left(\mathbf{y}_i[t]\right) & \text{if } \mathbf{x}_i[t] = \mathbf{s}_i[t] \\ q_{et}\left(\mathbf{y}_i[t]\right) & \text{if } \mathbf{x}_i[t] \neq \mathbf{s}_i[t]. \end{cases}$$

In the above formulation, all $g_t(\cdot)$, $q_{ct}(\cdot)$ and $q_{et}(\cdot)$ are discrete p.m.fs.

2.2.2 Model Estimation

Given data $\mathcal{R}$, the standard way to learn HMM parameters is to employ the iterative Expectation-Maximization (EM) algorithm [Rabiner (1989)]. However, our model is overparameterized. Many of the allowable transitions among $k$mers in the state space exist because of errors in the read set, but errors are handled by the emission distribution. To overcome this difficulty, we take advantage of the finiteness of genome $\mathcal{G}$. There should be exactly $|\mathcal{G}| - k + 1$ non-zero transition probabilities, $p_1(\cdot \mid \cdot)$, when *all* genomic $k$mers are unique, and even fewer in repetitive genomes. We regularize parameter estimation to reduce the number of non-zero transition probabilities.

For a $k$mer $\boldsymbol{\omega}$ observed in more than one context, the outgoing transition probabilities $\mathbf{p}_{\boldsymbol{\omega}} = \{p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}), \boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})\}$ lie in the 1-, 2-, or 3-simplex. To ensure a tractable M step, we assign the same label to *all* transitions from each $\boldsymbol{\omega}$ (§A.2.1). Thus, labeling transitions is equivalent to labeling $k$mers, and we let $\mathcal{L}(\boldsymbol{\omega})$ denote the label of $\boldsymbol{\omega}$. The following penalty, proposed in Candès et al. (2008); Armagan et al. (2013) and for probabilities in Alexander and Lange (2011), pushes $\mathbf{p}_{\boldsymbol{\omega}}$ toward a vertex or edge of the simplex according to the evidence supporting each observed transition.

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{\substack{\boldsymbol{\omega} \in \mathcal{K}, \boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega}), \\ \mathcal{L}(\boldsymbol{\omega}) = 1}} \log[1 + p_1(\boldsymbol{\nu}|\boldsymbol{\omega})/\gamma], \tag{2.5}$$

where γ is a positive constant and $\boldsymbol{\theta}$ is the vector of model parameters. The penalized log-likelihood we seek to maximize is

$$\begin{aligned}\ell(\boldsymbol{\theta} \mid \mathcal{R}) = -\rho\mathcal{J}(\boldsymbol{\theta}) \;\; + \\ \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{R}} \Bigg\{ \sum_{\boldsymbol{\omega}\in\mathcal{N}_{ik}(\mathbf{x}_{i,k})} \big[\log\mu(\widetilde{\boldsymbol{\omega}}) + \log f_k(\mathbf{x}_{i,k},\mathbf{y}_{i,k} \mid \boldsymbol{\omega})\big] + \\ \sum_{t=k+1}^{l} \sum_{(\boldsymbol{\omega},\boldsymbol{\nu})\in\mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \big[\log p_1(\boldsymbol{\nu}_{[k]}|\boldsymbol{\omega})\mathbb{1}_{\{\mathcal{L}(\boldsymbol{\omega})=1\}} + \log p_2(\boldsymbol{\nu}_{[k]}|\boldsymbol{\omega})\mathbb{1}_{\{\mathcal{L}(\boldsymbol{\omega})=2\}} \\ + \log f_t(\mathbf{x}_{i[t]},\mathbf{y}_{i[t]} \mid \boldsymbol{\nu},\mathbf{x}_{i,t}^{-})\big]\Bigg\},\end{aligned} \tag{2.6}$$

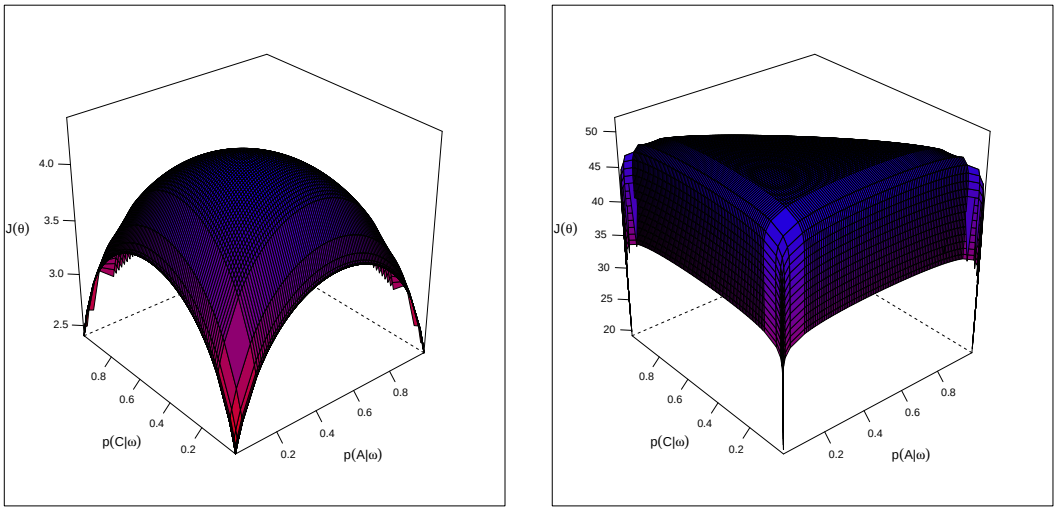
where $\mathcal{N}_{it}^{\otimes}(\mathbf{x}_i) = \{(\boldsymbol{\omega},\boldsymbol{\nu}) \in \mathcal{N}_{i,t-1}(\mathbf{x}_{i,t-1}) \times \mathcal{N}_{it}(\mathbf{x}_{i,t}) : \boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})\}$ and $\rho > 0$ is the regularization parameter. We employ the EM-algorithm to iteratively maximize (2.6) (for details, see §A.3.)

We briefly illustrate the effect of the penalty on transition probabilities from a $k$mer $\boldsymbol{\omega}$ with three observed transitions in $\mathcal{R}$, say $\boldsymbol{\omega} \to \{A, C, T\}$. In this case, $\mathbf{p}_{\boldsymbol{\omega}}$ lies in a 2-simplex, with $p_1(T \mid \boldsymbol{\omega}) = 1 - p_1(A \mid \boldsymbol{\omega}) - p_1(C \mid \boldsymbol{\omega})$. Fig. 2.4 shows the penalty function surface $\mathcal{J}(\mathbf{p}_{\boldsymbol{\omega}})$ against the two free parameters $p_1(A \mid \boldsymbol{\omega})$ and $p_1(C \mid \boldsymbol{\omega})$ for $\gamma = 10^{-8}$. As $\gamma \to 0$, the penalty becomes ℓ_0-like such that transitions parameters are effectively *thresholded*, driven to 0 when small enough and left untouched otherwise. In fact, the parameter ρ can be interpreted as the threshold value for the expected number of $\boldsymbol{\omega} \to b$ transitions (§A.3.4.) Above the threshold, the maximum penalized-likelihood estimator of $p_1(b \mid \boldsymbol{\omega})$ is close to its maximum likelihood estimate, and below the threshold, the estimate is essentially zero. Under the $k$mer-uniqueness assumption, transitions created by errors literally do not exist in the genome and should be driven to 0, so we use a small γ and choose ρ to identify likely true transitions (more on ρ in §A.5.1.)

2.2.3 Implementation Issues

Several implementation details play a crucial role in the performance of PREMIER. We highlight these issues here and refer to the reader to an in-depth discussion in the Supplementary Information.

- The labeling of $k$mers can significantly affect error correction performance. We discuss the reason and an algorithm for labeling the $k$mers that results in effective error correction.



(a) (b)

Figure 2.4: The penalty function $\mathcal{J}(\mathbf{p}_{\boldsymbol{\omega}})$ surface for $\gamma = 10^{-1}$ and $\gamma = 10^{-8}$.

Changing γ changes the curvature of the surface, but the penalty clearly prefers $\mathbf{p}_{\boldsymbol{\omega}}$ on the simplex edges, especially the vertices.

- We derive the iterative penalized-EM algorithm and present an algorithm for solving the nonlinear equations in the M step. We also discuss the ascent property of the objective function across the iterations.

- The penalized EM procedure operates by driving the erroneous transition probabilities aggressively to zero. In addition, our emission distribution is constrained to emit sequences close in the Hamming metric to the true state sequence. Thus, over the course of the iterations it is possible that the likelihood of certain reads is driven to zero. We discuss how this issue is handled in §A.3.5.2.

- We present a method for choosing k using an estimate of the genome length and information from the read set. We also discuss the robustness of our method to choice of k.

2.3 Results

To measure the performance of PREMIER relative to other state-of-the-art methods, we corrected errors in multiple real Illumina sequencing datasets with varying coverage levels. For

comparison, we selected publicly available error correction methods from the three dominant classes of methods [Molnar and Ilie (2014)]. Specifically, we compared the k-spectrum-based methods Musket (v1.0.7) [Liu et al. (2013)] and BLESS (v0.24) [Heo et al. (2014)], the suffix tree/array-based methods Fiona (`fiona_illumina` v0.2) [Schulz et al. (2014)] and HiTEC (v1.0.2) [Ilie et al. (2011)], and the multiple sequence alignment-based Coral (v1.4) [Salmela and Schröder (2011)] and Karect [Allam et al. (2015)]. An additional five methods are discussed in the §A.5.2.7. All error correction methods were benchmarked against an obtained ground truth.

2.3.1 Datasets & methods

We prepared four datasets for evaluation, all from real Illumina sequencing experiments (see Table 2.1 for details). Despite sharing similar overall error rates, $\boldsymbol{e}$, the four datasets have noticeable differences, in genome length $|\mathcal{G}|$, coverage, and genome complexity. To measure genome complexity in terms of repetitive elements (including tandem repeats, satellite DNA, long interspersed repeats, and low complexity regions, in RepeatMasker terminology), we have computed the percentage of genome masked out by RepeatMasker [Smit et al. (1999)], denoted as $R\%$, for every reference genome (or chromosome). All datasets, except $D4$, contain a considerable fraction of repetitive elements, and thus are good candidates to test model robustness to violation of the $k$mer-uniqueness assumption.

Datasets $D1$–$D3$ are standard Illumina pair-ended read sets with reference genomes (`WormBase WS238`, `flybase r6.02` and `GRCh37`) where ground truth errors are identified using sequence alignment, whereas dataset $D4$ is obtained via duplex sequencing (§A.6.2.)

Since error correction is more challenging when coverage is low, we simulated scenarios where the same references would be sequenced with much fewer reads ($5\times$ and $10\times$ coverage for $D1$-$D3$, and $5\times$–$50\times$ for $D4$) by randomly subsampling reads from the original datasets. To reduce noise in the results, the subsampling was replicated five times for $D1$–$D3$, thirty times for $D4$ at each coverage level, and performance measures were averaged.

All error correction methods were run per published recommendations or instructions in the software manual. Both Fiona and Coral can correct insertion/deletion errors in addition

Table 2.1: Illumina datasets used for performance evaluation.

Dataset	Source	Library	Coverage	e (in %)	Reference	$\|\mathcal{G}\|^a$	$R\%^b$
D1	*C. elegans chr.I* SRR065390	2×100bp	54× (10×, 5×)	0.58	WormBase WS238	15,072,434	14.48%
D2	*D. melanogaster* SRR492060	2×76bp	21× (5×)	0.47	flybase r6.02	143,725,995 (142, 573, 017)	28.61%
D3	*H. sapiens chr.14* [Salzberg et al. (2012)]	2×101bp	30× (5×)	0.89	GRCh37 NC000014.8	107,349,540 (88, 289, 540)	41.61%
D4	*H. sapiens* mitochondria [Schmitt et al. (2012)] [d]	2 × 80bp[c]	171, 415× (50 × −5×)[e]	0.36	GRCh37 (hg19)	16,569	2.52%

[a] Proportion of genome comprised of tandem repeats, satellite DNA, long interspersed repeats, and low complexity regions as computed by RepeatMaster [Smit et al. (1999)].
[b] Numbers in parentheses are reference genome lengths after removing "N" bases.
[c] 2×80 bp after removing barcodes from the original reads, which are of length 2×101bp.
[d] Sequencing data for $D4$ was provided by the authors of Schmitt et al. (2012).
[e] $(50\times, 40\times, 30\times, 20\times, 15\times, 10\times, 5\times)$

to substitution errors, and were run in "Illumina mode," correcting only substitutions, as provided by the software. In addition, we extensively tuned those parameters known to affect performance of competing methods with lower performance and report the optimal performance (§A.5.)

2.3.2 Performance comparisons

Although others have categorized error correction methods into three or more classes [Molnar and Ilie (2014); Yang et al. (2013)], we find two classes more useful for explaining performance outcomes. The k, d-local methods detect an error by considering, at most, the k $k$mers straddling the error, the counts in read set $\mathcal{R}$ of all $k$mers within some maximum Hamming distance d of these $k$mers, and the quality scores in the $2k - 1$ window centered on the error. All $k$mer-based methods and suffix-tree/array-based methods with k set to the maximum suffix length are k, d-local methods. The read-local methods use the entire read to make a decision. All error correction methods, except the alignment-based methods Coral and Karect, are k, d-local methods. Clearly, the read-local methods have the potential to beat any k, d-local method.

For all datasets, the availability of ground truth errors allows us to evaluate the quality of error correction by treating it as a binary classifying process and determining the number of true

Table 2.2: Error correction performance on Illumina sequencing datasets.

		BLESS			Coral			Fiona			HiTEC			Karect			Musket			PREMIER		
Dataset		sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p
D1	54×	0.802	0.793	0.10	0.747	0.592	0.19	0.841	0.822	0.08	0.847	0.774	0.11	0.931	0.925	0.04	0.834	0.821	0.08	**0.931**	**0.926**	**0.04**
	10×	0.710	0.699	0.14	0.681	0.487	0.24	0.757	0.698	0.14	0.740	0.281	0.34	0.885	0.870	0.06	0.770	0.725	0.13	**0.900**	**0.880**	**0.06**
	5×	0.542	0.503	0.23	0.593	0.298	0.33	0.638	0.524	0.22	0.590	-0.347	0.63	**0.807**	**0.747**	**0.12**	0.679	0.606	0.19	0.806	0.714	0.13
D2	21×	0.708	0.680	0.19	0.655	0.389	0.36	0.763	0.679	0.19	0.758	0.620	0.22	0.822	0.788	0.13	0.744	0.681	0.19	**0.828**	**0.800**	**0.12**
	5×	0.570	0.519	0.28	0.616	0.328	0.39	0.651	0.511	0.29	0.635	0.328	0.39	0.742	0.655	0.20	0.650	0.579	0.25	**0.763**	**0.661**	**0.20**
D3	30×	0.739	0.714	0.25	0.658	0.464	0.48	0.846	0.800	0.18	0.845	0.642	0.32	**0.942**	**0.924**	**0.07**	0.822	0.777	0.20	0.934	0.912	0.08
	5×	0.503	0.448	0.49	0.502	0.361	0.57	0.645	0.564	0.43	0.654	0.123	0.86	**0.805**	**0.736**	**0.23**	0.695	0.594	0.36	0.785	0.687	0.28

positives (TP), false negatives (FN), false positives (FP) and true negatives (TN). Following this we compute the metrics sensitivity (sn.) $= \frac{TP}{TP+FN}$ and gain $= \frac{TP-FP}{TP+FN}$.

We evaluated the methods on 23 alignment-based datasets ($D1, D2, D3$ and five replicates each of $D1_{10\times}$, $D1_{5\times}$, $D2_{5\times}$ and $D3_{5\times}$) (Table 2.2). Among k, d-local methods, PREMIER is the top performer on both sn. and gain. Furthermore, as (sn. − gain) is proportional to the number of false positives, PREMIER is clearly the most conservative method, introducing far fewer new errors than competing methods. In terms of gain, PREMIER outperforms the runner-up by 10.4% (Fiona) on $D1_{54\times}$, 11.9% (Musket) on $D2_{21\times}$, and 11.2% (Fiona) on $D3_{30\times}$. When compared with read-local alignment methods, PREMIER slightly edges Karect on datasets $D1$ and $D2$, but is 1.2% inferior to Karect on $D3$.

The relatively weaker performance on $D3$ for PREMIER is attributable to our constraints on model complexity. To adaptively determine the Hamming distance parameter d_{it}, PREMIER controls the maximum neighborhood sizes, which are fixed quantities (§A.2.2.) Since the neighborhoods are considerably larger for $D3$ due to its highly repetitieve genome, the average d_{it} become smaller, leading to limited error discovery capacity. This proposition is supported by our empirical experiments, which showed that when increasing k from 24 to 30 on $D3$, the gain metric improves by 3%. As larger $k$mer size induces sparser neighborhoods, we expect PREMIER's performance to further improve for $k > 30$.

Low coverage datasets are difficult to correct for all error correction methods. Both BLESS and Musket performed poorly on datasets with 5× coverage (negative gains on $D2_{5\times}$ and $D3_{5\times}$), unless we manually adjust their $k$mer coverage cut-off values (§A.5.) With the few exceptions of Musket on $D2_{5\times}$ and $D3_{5\times}$, PREMIER outmatches other k, d-local methods, plus Coral,

Table 2.3: Error correction performance on duplex-sequencing dataset $D4$.

		BLESS			Coral			Fiona			HiTEC			Musket			PREMIER		
Dataset		sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p	sn.	gain	e_p
D4	5×	0.797	0.787	0.08	0.831	0.822	0.07	0.818	0.804	0.07	0.737	0.731	0.10	0.735	0.717	0.10	**0.865**	**0.836**	**0.06**
	10×	0.923	0.910	0.03	**0.960**	**0.944**	**0.02**	0.931	0.916	0.03	0.886	0.874	0.05	0.946	0.932	0.03	0.959	0.943	0.02
	15×	0.953	0.940	0.02	0.978	0.964	0.01	0.960	0.944	0.02	0.946	0.933	0.03	0.972	0.959	0.02	**0.981**	**0.967**	**0.01**
	20×	0.964	0.952	0.02	0.975	**0.962**	**0.01**	0.958	0.943	0.02	0.943	0.93	0.03	0.971	0.959	0.02	**0.976**	**0.962**	**0.01**
	30×	0.975	0.961	0.01	0.981	0.967	0.01	0.966	0.949	0.02	0.954	0.941	0.02	0.979	0.965	0.01	**0.982**	**0.968**	**0.01**
	40×	0.975	0.963	0.01	0.978	0.966	0.01	0.967	0.951	0.02	0.953	0.941	0.02	0.977	0.965	0.01	**0.980**	**0.968**	**0.01**
	50×	0.973	0.961	0.01	0.978	0.964	0.01	0.966	0.948	0.02	0.952	0.939	0.02	0.975	0.963	0.01	**0.979**	**0.966**	**0.01**

with double-digit gaps of gain on low-coverage datasets. The comparison with Karect is more competitive, with PREMIER came first on $D1_{10\times}$ and $D2_{5\times}$, but fell short on $D1_{5\times}$ and $D3_{5\times}$.

For the duplex-sequencing based dataset $D4$, we repeated the data analysis procedures documented in the supplementary material of Schmitt et al. (2012) to identify the ground truth errors (see SI). Next, we produced a series of reduced coverage datasets (5×, 10×, 15×, 20×, 30×, 40×, 50×) by randomly subsampling $D4$. At each coverage level, 30 random samples were created and analyzed. Table 2.3 reports the mean error correction performance, at each coverage level.

All methods yield similar performance, due to the low genome complexity. Despite the small margin between methods, PREMIER remains superior in removing errors, especially in low-coverage scenarios. At 10× coverage, Coral marginally beats PREMIER's gain, but at 5× coverage level, PREMIER removes 1.4% more errors than Coral, and more than 3% more errors than all other methods. We note that Coral is not a consistent performer, ranking among the worst on all alignment-based datasets, and its performance can be highly sensitive to run parameters (§A.5.)

2.4 Discussion

In this article, we presented PREMIER, a flexible, fully-probabilistic method for correcting errors in Illumina sequencing. PREMIER is a k, d-local method that statistically models the sequencing process as a Hidden Markov model. Utilizing $k$mers as the states of the HMM, the transitions between $k$mers is a k-th order Markov chain that captures the local genomic dependence. The HMM emits the observed nucleotides and quality scores, while account for error characteristics of the sequencer.

In our results analysis, PREMIER consistently outperforms other state-of-the-art k, d-local methods, and stays competitive with Karect, the read-local alignment-based method. We attribute much of PREMIER's success to its explicit statistical modeling of genomic dependence and sequencer error characteristics, on top of which, a few unique modeling features of PREMIER should also be noted.

We tackled the overparameterization in our model by regularizing it with the ℓ_0-type penalty (2.5), controlled by the shape parameter γ, and the cut-off parameter ρ. We fix γ at a small value to seek sparse solution in the parameter space, by driving small transition probabilities to zero. The threshold value ρ is similar to the $k$mer coverage cut-off used in Liu et al. (2013); Heo et al. (2014); Ilie and Molnar (2013), with two notable differences. First, we used more conservative value of ρ so that fewer valid transition probabilities are mistakenly driven to zero due to coverage variation. Second, unique transitions are preserved under constraint (3.1) even if their expected number of transition is below threshold.

Whereas individual reads are single-stranded sequences, DNA molecules are inherently double-stranded. To exploit the dependence within the two strands, we developed a $k$mer labeling strategy, which assigns half of the $k$mers label *1* to mark their transition probabilities as free parameters, or label *2* for dependent parameters. Heuristic labeling algorithms were devised to optimize the sparsifying effect of the penalty, by preferably assigning label *1* to $k$mers with non-unique transitions, which, under the $k$mer uniqueness assumption, localize the errors. Despite the conceptual similarity, the labeling of $k$mer does not imply strandedness. Indeed, some $k$mers may exist on both strands and thus have undetermined strandedness. In particular, since we use even k, a subset of $k$mers in $\mathcal{K}$ satisfies $\boldsymbol{\omega} = \overline{\boldsymbol{\omega}}$. These can be treated as special cases of $k$mer non-uniqueness, and does not affect the model validity, since we do not require $\mathcal{L}(\boldsymbol{\omega}) \neq \mathcal{L}(\overline{\boldsymbol{\omega}})$.

One major challenge to apply Hidden Markov model to error-correction is the model tractability for large datasets, since estimating a swarm of model parameters requires calculations over complex state paths, and may bear computational overhead that exceeds memory and time constraints. To curb the model complexity, the hidden states are restricted to be within a small neighborhood of the observed $k$mer, where the neighborhood sizes are bounded

by a read- and position-dependent Hamming distance, $d_{i,t}$, which is adaptively determined in response to characteristics like quality scores and $k$mer uniqueness.

To model the generative process of the sequencer, PREMIER always initiate the underlying Markov chain at the first position of the read, and progresses in a 5' to 3' direction. Most other $k$mer-based methods have the freedom to start error correction at virtually any position, on either strand of the read. In addition, the construction of neighborhoods $\mathcal{N}_{it}(\mathbf{x}_{i,t})$ for $t > k$ is dependent on the first one, $\mathcal{N}_{ik}(\mathbf{x}_{i,k})$. As a result, PREMIER's performance is more sensitive to errors within the first $k$mer. Our heuristic rule for constructing $\mathcal{N}_{ik}(\mathbf{x}_{i,k})$ may fail in scenarios like low-coverage, non-uniform coverage, and repetitive regions. Indeed, our experiments showed that PREMIER's performance excels in reads with clean first $k$mer, but languishes otherwise. To remedy the underwhelming performance in reads with erroneous first $k$mers, we plan to utilize multiple-sequence alignments for constructing the first neighborhoods.

The statistical modeling approach we pose can be easily extended for further development. The emission distribution can be improved to model complex context-dependent errors [Nakamura et al. (2011)] or, better still, to directly model the raw fluorescent intensities of Illumina sequencers, essentially turning PREMIER into an error-aware base-caller. We are also working to incorporate insertion and deletion errors by extending the state space with specialized states, to make PREMIER widely applicable to other NGS platforms.

CHAPTER 3. PREMIER-BC: INTEGRATED ILLUMINA BASE-CALLING AND ERROR-CORRECTION APPROACH USING HIDDEN MARKOV MODEL

Abstract

Next-generation sequencing errors are largely avoided or removed in two consecutive stages in the bioinformatics data analysis pipeline, base-calling followed by error-correction. Interestingly, base-callers rarely utilize information available about the underlying genome sequence, whereas error-corrections methods seldom utilize properties of the sequencing machine. The data left unexploited in either stage ultimately leads to suboptimal error removal capability, even when base-calling and error correction are sequentially applied. Inspired by Das and Vikalo (2013), who proposed an Hidden Markov model (HMM) for Illumina base-calling, yet without modeling the sequence dependence in the genome, in this work, we develop PREMIER-bc, a unified base-caller and error corrector that utilizes a HMM to directly model the sequencing-by-synthesis process of Illumina sequencers. PREMIER-bc builds on top of PREMIER, our previous HMM for Illumina error correction, and continues to use the transition distribution to model non-random genome structures, but introduced extensions to the emission distribution, so that the error properties during the generation of fluorescent intensities can be properly modeled. Our results on Φ-X174 dataset demonstrate significant improvements in base-calling quality over existing base-callers, and more importantly, the sequential application of base-calling and error correction.

3.1 Introduction

The unabating development of sequencing platforms has made significant inroads in improving the cost-efficiency and throughput of sequencing nucleic acid sequences [van Dijk et al. (2014)]. Although the cost-effective and high-throughput appeal of the next-generation sequencing technologies have catalytically propelled biological research, challenges remain in acquiring accurate, high-quality data from NGS platforms. All high-throughput sequencers generate data with error rates significantly higher than Sanger sequencing [Quail et al. (2012b), Loman et al. (2012), Ross et al. (2013)], and such impurity is known to interfere with many downstream analyses [Alic et al. (2016)].

In light of the imperative to improve sequencing data quality, plentiful solutions targeting different stages of a sequencing experiment have been offered. Some approaches adopt customized library preparation [Schmitt et al. (2012), Lou et al. (2013)] or PCR amplification [Quail et al. (2012a)] to control sequencing bias or errors. Another highly versatile option is to attenuate noise via bioinformatics analysis. We focus on two early phases of bioinformatics pipelines for sequencing analysis: base-calling and error correction. A base-caller converts, in massive parallelism, platform-specific raw sequencer signals into reads of a standard format, via explicit modeling of the sequencing process and its error properties. Error correction, sometimes applied after base-calling, refines the output of base-callers by exploiting the redundancy in the high-throughput reads.

Notwithstanding modularity and flexibility, there are some non-trivial downsides to stacking base-callers and error-correctors to create bioinformatics pipelines. In order to handle the deluge of data, base-calling algorithms usually conduct data conversion in real-time, quickly followed by deletion of the unwieldy raw signals, so they can rarely tap into the structural signal of the underlying genome. To exemplify, consider the widely used Illumina platform. According to recent surveys of Illumina base-calling methods by Ledergerber and Dessimoz (2011) and Cacho et al. (2015), most of the existing base-callers [Rougemont et al. (2008), Kao et al. (2009), Bravo and Irizarry (2010), Massingham and Goldman (2012), Das and Vikalo (2013)] build on similar statistical models, with common components to model well-studied error characteristics in

Illumina's fluorescent intensity data, including cross-talk, phasing/prephasing and signal decay, but make no use of the genome content. The remaining few machine learning based methods [Erlich et al. (2008), Kircher et al. (2009), Renaud et al. (2013)] utilize reference genomes, but only as control samples (*e.g.* the ϕ-X174 genome) to train their model via supervised learning. The features used for base-calling still consist of only a few cycles of intensities. On the other hand, downstream error correction methods make simplifying assumptions about sequencing errors and often discard forwarded information (quality scores) from the base calling process all together [Laehnemann et al. (2015)].

The information left unutilized by either tool or lost when transitioning between tools in the pipeline can be recouped by combining the two applications into a single, unified model. The same idea underlies TotalRecaller [Menges et al. (2011)], a base-caller that aligns Illumina data to a provided reference genome. However, its reliance on a known reference inevitably introduces bias when the presumptive reference sequence is different from the target genome being sequenced, which is true for any noteworthy sequencing experiment, and makes TotalRecaller inappropriate for *de novo* sequencing experiments.

In this article, we present PREMIER-bc, an integrated approach to produce accurate Illumina reads by unifying base-calling and error-correction in one probabilistic framework. The key contribution of this work is to formulate a Hidden Markov model (HMM) that closely resembles the operation of Illumina sequencers. Our model is similar to, and was partly inspired by, the HMM proposed by Das and Vikalo (2013), but with notable differences. Critically, PREMIER-bc utilizes $k$mers, whereas Das and Vikalo (2013) use nucleotide dimers (AA, AC, ..., TT) as the state space of their HMM. To learn the complex hidden state space, *i.e.* the genome, PREMIER-bc is necessarily an off-line base-caller, and utilizes the output of a basic base-caller, like Bustard, to construct the plausible hidden state space. The emission distribution of PREMIER-bc, which models the generation of observed intensities, derives from a recurring base-calling model reviewed in [Cacho et al. (2015)], but with simplifications to allow for computational tractability. Replacing dimers in the state space with $k$mers leads to an over-parameterized model, since many $k$mers are propagated by sequencing errors and do not exist

in the genome. To address this issue, we regularize the model estimation with a sparsifying ℓ_0-type penalty.

Our analysis of base-calling quality on a ϕ-X174 dataset demonstrates significant reduction in error rate over publicly-available Illumina base-callers. By jointly base calling and error correcting Illumina intensity data, PREMIER-bc also outperforms the sequential application of base-caller (Bustard) and error-corrector (PREMIER), underscoring the appeal of a unifying statistical model over conventional pipelining of bioinformatics tools.

3.2 Methods

In this section, we describe our model, PREMIER-bc, that unifies Illumina base-calling and error-correction. Given a target genome to be sequenced, we denote its forward and reverse strands as $\mathcal{G}$ and $\overline{\mathcal{G}}$ respectively. Each strand is a string of length $|\mathcal{G}|$ over the alphabet $\Omega = \{A, C, G, T\}$. The sequencing process starts by randomly breaking copies of $\mathcal{G}$ and $\overline{\mathcal{G}}$ into short fragments, which are collectively referred to as $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$. The i-th fragment $\mathbf{s}_i$ is then sequenced and produces raw fluorescent intensities $\boldsymbol{I}_i$, which is in turn converted by a base-caller into the i-th read $(\mathbf{x}_i, \mathbf{y}_i)$. We assume that each read is of fixed length l, and consists of nucleotides (base-calls) $\mathbf{x}_i$ and quality scores $\mathbf{y}_i$.

Regardless of the sources of input data, the common objective of base-calling and error-correction is to recover the latent genomic sequences $\mathcal{S}$ with high accuracy. The principal idea embedded in our probabilistic framework for base-calling and error correction, is to model observed data for each read as an independent emission from a Hidden Markov model (HMM). We model the hidden states of the HMM, which are $k$mers, as a Markov chain of order k to harness the local dependence in the genome. The emission distribution is utilized to model the generating process of observed data from the sequencer and upstream data analysis, and is designated to be flexible to adapt to the nuances between applications and platforms.

For the ease of presentation, we first review the HMM for error-correction proposed in chapter 2. The model, PREMIER, serves as a "base" framework, on top of which model extensions are proposed to address the base-calling problem. In addition to the notation presented above, the following will be used throughout this article. Let $\mathbf{s}_i$ denote the true genomic sequence

that generates $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{R}$. Given $\mathbf{s}_i$, we define $\mathbf{s}_{i[j]}$ as the j-th nucleotide of $\mathbf{s}_i$, and define $\mathbf{s}_{i,t} \equiv \mathbf{s}_{i[t-k+1\ldots t]}$ as the $k$mer ending at the t-th nucleotide, which is the substring from position $t-k+1$ to t, both termini included.

Any sequence of length l can be decomposed into $l-k+1$ overlapping $k$mers, *e.g.* $\mathbf{s}_i = \mathbf{s}_{i,k}, \mathbf{s}_{i,k+1}, \ldots, \mathbf{s}_{i,l}$, where two adjacent $k$mers $\mathbf{s}_{i,t}$ and $\mathbf{s}_{i,t+1}$ share a common $k-1$ suffix/prefix. The juxtaposition of two overlapping $k$mers $\boldsymbol{\omega}$ and $\boldsymbol{\nu}$ with $\boldsymbol{\omega}_{[2\ldots k]} = \boldsymbol{\nu}_{[1\ldots k-1]}$ is also referred to as a transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$. For any given $k$mer $\boldsymbol{\omega}$, let $\overline{\boldsymbol{\omega}}$ denote its reverse complement.

3.2.1 A Hidden Markov modeling approach to error correction

In the context of error correction, the observed data is $\mathcal{R}$, the read set generated by the sequencer and base-caller duo. To identify the sequencing errors in read $(\mathbf{x}_i, \mathbf{y}_i)$, PREMIER models $(\mathbf{x}_i, \mathbf{y}_i)$ as an independent emission from a Hidden Markov model, with the following components.

State space. If $\mathcal{R}$ is generated by an HMM, then an erroneous $k$mer is correctable only if its true, generating $k$mer is in the state space, denoted by $\mathcal{K}$. We let $\mathcal{K}$ be the set of all observed $k$mers in $\mathcal{R}$, plus their reverse complements. When the target genome is sequenced with decent coverage, it is reasonable to assume that $\mathcal{K}$ include all true genomic $k$mers.

Transition distribution By modeling the hidden states as a Markov chain of order k, the transition probability $p(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = 0$ if $\boldsymbol{\omega}_{[2\ldots k]} \neq \boldsymbol{\nu}_{[1\ldots k-1]}$, for $\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{K}$. The $k-1$ suffix-prefix match requirement allows us to parameterize the transition matrix as a $|\mathcal{K}| \times 4$ matrix, and use $p(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ and $p(b \mid \boldsymbol{\omega})$ interchangeably, where $b = \boldsymbol{\nu}_{[k]} \in \Omega$. To further sparsify the transition matrix, and to avoid interminging contexts in the genome, given $\boldsymbol{\omega} \in \mathcal{K}$, define $\mathcal{T}(\boldsymbol{\omega})$ to be set of $\boldsymbol{\nu} \in \mathcal{K}$ such that $\boldsymbol{\omega} \to \boldsymbol{\nu}$ or $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$ is observed in $\mathcal{R}$. Transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ is disallowed if $\boldsymbol{\nu} \notin \mathcal{T}(\boldsymbol{\omega})$. And we have,

$$\sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} p(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = 1, \forall\, \boldsymbol{\omega} \in \mathcal{K}. \tag{3.1}$$

Double-stranded dependence To model the inherent double-strandedness of genomic DNA, consider complementary $k$mers $\boldsymbol{\omega}, \overline{\boldsymbol{\omega}} \in \mathcal{K}$. Let $\widetilde{\boldsymbol{\omega}} \equiv \min\{\boldsymbol{\omega}, \overline{\boldsymbol{\omega}}\}$ in lexical order denote the

canonical $k$mer for the pair. We have $\mu(\widetilde{\boldsymbol{\omega}}) = \mu(\boldsymbol{\omega}) = \mu(\overline{\boldsymbol{\omega}})$ under the assumption of uniform sequencing coverage. Furthermore, if $\boldsymbol{\omega} \to \boldsymbol{\nu}$ is unique in $\mathcal{G}$, it implies $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$ is also unique in $\overline{\mathcal{G}}$ and

$$p_2(\overline{\boldsymbol{\omega}}_{[k]} \mid \overline{\boldsymbol{\nu}}) = \frac{\mu(\widetilde{\boldsymbol{\omega}}) \cdot p_1(\boldsymbol{\nu}_{[k]} \mid \boldsymbol{\omega})}{\mu(\widetilde{\boldsymbol{\nu}})}, \tag{3.2}$$

where the transitions are now labeled by indicators $\{1, 2\}$ to indicate *free* or *dependent* parameters of the model, thus reducing the number of free parameters by half, effectively doubling the coverage. Eq. (3.2) holds even if we relax the uniqueness assumption, and thus is true for *any* $\boldsymbol{\omega} \to \boldsymbol{\nu}$ and $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$. All we need is a consistent method to assign labels to the transitions such that transitions $\boldsymbol{\omega} \to \boldsymbol{\nu}$ and $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$ always carry opposite labels, see Supplementary Information (§A.2.1) for details.

Emission distribution The emission distribution in PREMIER models the *residual* error characteristics of the sequencer and base-caller duo, including position-dependent errors, residual cross-talk effects and quality scores. We omitted the definitions here for brevity.

3.2.2 Integration of base-calling and error-correction

PREMIER-bc integrates base-calling and error-correction by combining data from two sources: the error properties learnt from the sequencers' raw intensities, and the local nucleotide dependence from the underlying genome. We note that transitioning from error correction to base-calling does not change the underlying sequencing process, hence we can directly inherit the state space and the transition distribution from PREMIER without modifications. However, the observed data in PREMIER-bc becomes the set $\mathcal{I} = \{\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_n\}$, where each $\boldsymbol{I}_i$ is a $l \times 4$ matrix of fluorescence intensities observed for sequence i across all l positions, In particular,

$$\underset{l\times 4}{\boldsymbol{I}_i} = \begin{bmatrix} \boldsymbol{I}'_{i,1} \\ \vdots \\ \boldsymbol{I}'_{i,l} \end{bmatrix},$$

where $\boldsymbol{I}'_{i,t} = \left(I^A_{i,t}, I^C_{i,t}, I^G_{i,t}, I^T_{i,t}\right)$ denotes the intensities of the four (A, C, G, T) channels in sequence i, at position t.

To be consistent with the terminology used in other base-calling articles, in what follows, we use "sequence" and "cluster" interchangeably. A sequence is also called a "cluster", because prior to sequencing, the single-molecule DNA templates are amplified with bridge amplification to form "clusters", so that the fluorescence can be bright enough for reliable measurements. Similarly, we may use "position" and "cycle" in the place of each other. During each sequencing cycle, a fluorescent labeled deoxynucleoside triphohsphate (dNTP) is synthesized. The newly incorporated dNTP blocks further polymerization, until the fluorescent label is optically determined and cleaved. Naturally, the current position on the sequence progresses by one base.

The change from $\mathcal{R}$ to a more information-rich $\mathcal{I}$ demands a new emission distribution that explicitly models the generation of $\boldsymbol{I}_i$ from the DNA template $\mathbf{s}_i$, while accounting for the cycle-dependent noises. We now describe the components of the HMM of PREMIER-bc in detail.

State space

PREMIER-bc relies on Bustard, Illumina's built-in base-caller, to obtain the "crude" read set $\mathcal{R}$, from which the state space $\mathcal{K}$ can be constructed in the same manner as PREMIER. Theoretically, even in the absence of $\mathcal{R}$ base-called by Bustard, we can extract $\mathcal{R}$ from $\mathcal{I}$ developing a simplistic base-calling model similar to Bustard, or utilizing any third-party base-caller.

Transition distribution

The transition distribution in PREMIER-bc explicitly models the sequencing-by-synthesis process of Illumina sequencers. Normally, each transition between $k$mers with suffix-prefix match corresponds to a *cycle* in Illumina sequencing and an incorporation of a new nucleotide. Infrequently the sequencer may synthesize zero, or more than one base in a single cycle. These irregularities are referred to as the *phasing* and *prephasing* effects, and induce insertion and deletion errors in $\mathcal{R}$. It is possible, via proper model extensions, for our model to handle phasing and prephasing accordingly. However, since Illumina sequencer predominantly produces

substitution errors, we defer this extension to future works. Henceforth, we assume that there is no phasing or prephasing in sequencing, therefore the base synthesized and read at cycle t in cluster i is always $\mathbf{s}_{i[t]}$.

Emission distribution: Cross-talks, intensity decaying and residual effects

We utilize the emission distribution to model the common error properties of the Illumina sequencers, including cross-talk, decay and residual effects. Let $\mathbf{s}_{i[t]}$ denote the base read at cycle t. Define $\boldsymbol{e}_b,\ b \in \Omega$ as 4-dimensional unit vectors with 1 at the index determined by b, and 0 elsewhere (*e.g.* $\boldsymbol{e}_A = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}'$, and $\boldsymbol{e}_T = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}'$.) Then, given $\mathbf{s}_{i[t]}$, the emission distribution behaves as a linear filter given by

$$\mathbb{E}\left(\boldsymbol{I}_{i,1} \mid \mathbf{s}_i[1]\right) = \Lambda_{i,1}\boldsymbol{C}_t\boldsymbol{e}_{\mathbf{s}_i[1]}, \tag{3.3}$$

$$\mathbb{E}\left(\boldsymbol{I}_{i,t} \mid \mathbf{s}_{i[t]}, \boldsymbol{I}_{i,t-1}\right) = \Lambda_{i,t}\boldsymbol{C}_t\boldsymbol{e}_{\mathbf{s}_{i[t]}} + \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}, \tag{3.4}$$

where $\alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}$ corresponds to the fluorescent intensity that leaks from the previous cycle, $\boldsymbol{C}_t$ is the 4×4 cross-talk matrix, and $\Lambda_{i,t}$ denotes the total light intensity at cycle t in cluster i.

To model the decay of $\Lambda_{i,t}$, BayesCall [Kao et al. (2009)] considered the following stochastic process, with cycle-dependent decay parameter δ_t

$$\begin{aligned}\Lambda_{i,1} &\sim N(\mu_1, \sigma_1^2),\\ \Lambda_{i,t}|\Lambda_{i,t-1} &\sim N\left((1-\delta_t)\Lambda_{i,t-1}, (1-\delta_t)^2\Lambda_{i,t-1}^2\sigma_t^2\right), t > 1.\end{aligned}$$

Softy [Das and Vikalo (2013)], citing empirical evidence that $\Lambda_{i,t}$ exhibits little variation around the mean for $t > 1$, simplify the above model to

$$\begin{aligned}\Lambda_{i,1} &\sim N(\mu_1, \sigma_1^2),\\ \Lambda_{i,t} &= \Lambda_{i,1}\prod_{j=2}^{t}(1-\delta_j), t > 1.\end{aligned}$$

In PREMIER-bc, we use a cluster-specific parameter λ_i to describe the light intensity at the first cycle and model the decay similarly to Softy, *i.e.*

$$\Lambda_{i,1} = \lambda_i, \quad \Lambda_{i,t} = \lambda_i\prod_{j=2}^{t}(1-\delta_j). \tag{3.5}$$

As will be evident shortly, this further simplification allows us to estimate the model parameters efficiently using the conventional Forward-Backward algorithm for the HMM.

Lastly, like PREMIER, to make the model tractable, we must limit the plausible hidden state sequences $\mathbf{s}_i$ that could possibly emit $\boldsymbol{I}_i$. Given the stochasticity in $\boldsymbol{I}_i$, it is not trivial to evaluate the distance between $\mathbf{s}_i$ and $\boldsymbol{I}_i$. Nevertheless, with the read $(\mathbf{x}_i, \mathbf{y}_i)$ called by Bustard available, we can employ the concept of the $k$mer neighborhood in PREMIER to eliminate all $\mathbf{s}_i$'s that contain an excessive number of mismatches from $\mathbf{x}_i$. Briefly, the neighborhood of observed $k$mer $\mathbf{x}_{i,t}$ is defined by the set

$$\mathcal{N}_{it}(\mathbf{s}_{i,t}) = \{\boldsymbol{\nu} : \boldsymbol{\nu} \in \mathcal{K} \text{ and } D(\mathbf{x}_{i,t}, \boldsymbol{\nu}) < d_{it}\}, \tag{3.6}$$

which depends on a read- and cycle-specific maximum Hamming distance parameter d_{it}. The operator $D(\boldsymbol{\omega}, \boldsymbol{\nu})$ returns the Hamming distance between two $k$mers $\boldsymbol{\omega}$ and $\boldsymbol{\nu}$, which is the number of mismatches between them.

All above considerations give rise to the following emission distribution:

- For $t = 1$,

$$\boldsymbol{I}_{i,1}|\mathbf{s}_{i,k}, \mathbf{x}_{i,k} \sim \mathbb{1}\{\mathbf{s}_{i,k} \in \mathcal{N}_{ik}(\mathbf{x}_{i,k})\} \times N\left(\Lambda_{i,1}\boldsymbol{C}_1\boldsymbol{e}_{\mathbf{s}_{i[1]}}, \|\Lambda_{i,1}\boldsymbol{e}_{\mathbf{s}_{i[1]}}\|_2^2\boldsymbol{\Sigma}_1\right). \tag{3.7}$$

- For $t > 1$,

$$\begin{aligned}\boldsymbol{I}_{i,t}|\boldsymbol{I}_{i,t-1}, \mathbf{s}_{i,t^*}, \mathbf{x}_{i,t^*} \sim\ & \mathbb{1}\{\mathbf{s}_{i,t^*} \in \mathcal{N}_{it}(\mathbf{x}_{i,t^*})\}\times \\ & N\left(\Lambda_{i,t}\boldsymbol{C}_t\boldsymbol{e}_{\mathbf{s}_{i[t]}} + \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}, \|\Lambda_{i,t}\boldsymbol{e}_{\mathbf{s}_{i[t]}}\|_2^2\boldsymbol{\Sigma}_t\right),\end{aligned} \tag{3.8}$$

  where $t^* = \max(t, k)$ is used to cope with the fact that hidden states in the HMM are $k$mers.

In the following text, we will use $g_1(\boldsymbol{I}_{i,1}|\mathbf{s}_{i,k}, \mathbf{x}_{i,k})$ and $g_t(\boldsymbol{I}_{i,t}|\boldsymbol{I}_{i,t-1}, \mathbf{s}_{i,t^*}, \mathbf{x}_{i,t^*})$ to denote the pdfs of the emission distribution.

3.2.3 Model estimation

We propose a two-stage algorithm to estimate the emission and transition parameters separately. Based on the setup of our model, the two stages can be roughly viewed as "base-calling" and "error-correction" processes.

Unlike transition probabilities, which model the genome structures, emission distribution parameters reflect the properties of the Illumina sequencer, and thus require more considerations. For Illumina sequencing, the DNA molecules are deposited in arrays of nano-wells called "flow cells". Each flow cell consists of multiple lanes, allowing different samples to be loaded and sequenced independently. Each lane is further divided into non-overlapping "tiles". During the imaging step at individual cycles, each tile produces a separate image of fluorescent luminance [Bentley et al. (2008)].

It is therefore reasonable to assume the emission parameters $(\boldsymbol{C}_t, \boldsymbol{\Sigma}_t, \alpha_t, \delta_t, \lambda_i)$ to be tile-specific, so that we can model the tile effects. Under this assumption, we can split the data into tiles, and fit our model to each tile individually. During the first stage of estimating emission parameters, we fix transition parameters that have been initialized from the reads. Next, conditioning on the estimated emission parameters, we lump *all* clusters across *all* tiles to estimate the transition probabilities with a penalized EM algorithm. Combining all data yields maximum coverage, which is crucial for the detection and subsequent removal of errors.

For ease of notation, it is convenient to partition the model parameters $\boldsymbol{\theta}$ into the transition part $\boldsymbol{\theta}_T$ and the emission part $\boldsymbol{\theta}_E = (\boldsymbol{\theta}_{E,1}, \boldsymbol{\theta}_{E,2}, \ldots, \boldsymbol{\theta}_{E,N})$, where N is the total number of tiles in the dataset. So, during the first stage, we condition on an initial estimate $\boldsymbol{\theta}_T^{(0)}$ of the transition parameters in order to estimate $\hat{\boldsymbol{\theta}}_E$. In the second stage, we condition on $\hat{\boldsymbol{\theta}}_E$ and estimate $\hat{\boldsymbol{\theta}}_T$.

3.2.3.1 Parameter initialization

Initializing the transition probabilities in PREMIER-bc is no different from the initialization procedure used in PREMIER. Therefore, we refer the readers to the Supplementary Information (§A.3.3) for more details, and focus on the initialization of emission parameters here.

For the signal decay and residual effect parameters, we use the following simple initial values

$$\lambda_i = 1, \quad \delta_t = 0, \qquad \alpha_t = 0, \quad \forall t. \tag{3.9}$$

Next, given $\bar{\mathcal{I}}$, we assume that, in every cluster, the first base-call produced by Bustard, $\mathbf{x}_{i,k[1]}$ is accurate. Under this assumption, and plugging-in $\lambda_i = 1, \alpha_t = 0, \delta_t = 0$ into Eq. (3.31) at $t = 1$, we get the following simple form

$$\boldsymbol{c}_1^{b\,(0)} = \frac{\sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \mathbb{1}\{\mathbf{x}_{i,k[1]} = b\} \boldsymbol{I}'_{i,1}}{M}. \tag{3.10}$$

It follows, if we let $\boldsymbol{c}_t^{b\,(0)} = \boldsymbol{c}_1^{b\,(0)}, \forall t > 1$, all cross-talk matrices $\boldsymbol{C}_t$ are known.

Using the initial values of $\boldsymbol{C}_1$, we can use the formula in Eq. (3.35) to compute the initial covariance matrix ${\boldsymbol{\Sigma}_1}^{(0)}$. Similarly, we then assume that ${\boldsymbol{\Sigma}_t}^{(0)} = {\boldsymbol{\Sigma}_1}^{(0)}$ for all $t > 1$.

3.2.3.2 Model estimation framework

In this section, we discuss the procedure to estimate the emission parameters $\boldsymbol{\theta}_{E,j}$, specifically for tile j. We assume the tiles are independent, hence we can partition the intensity data $\mathcal{I}$, as well as the reads $\mathcal{R}$, into tile sets $\mathcal{I} = \{\mathcal{I}^1, \ldots, \mathcal{I}^N\}$, $\mathcal{R} = \{\mathcal{R}^1, \ldots, \mathcal{R}^N\}$. Let $N_j = |\mathcal{I}^j|$ denote the number of clusters in tile j. Since N_j is usually large, we can accelerate the estimation by randomly sampling M_j clusters to comprise the subset $\bar{\mathcal{I}}^j$. Then, the tile-specific parameters $\boldsymbol{\theta}_{E,j} = \{\boldsymbol{C}_t^j, \delta_t^j, \alpha_t^j, \boldsymbol{\Sigma}_t^j\} \cup \{\lambda_1^j, \ldots, \lambda_{M_j}^j\}$ can be estimated through the maximization of the log-likelihood $\ell\left(\boldsymbol{\theta}_{E,j} \mid \bar{\mathcal{I}}^j; \boldsymbol{\theta}_T^{(0)}\right)$. However, since λ_i^j are cluster-specific, for clusters $\boldsymbol{I}_i \in \mathcal{I}^j \setminus \bar{\mathcal{I}}^j$, the estimation of λ_i^j will be deferred to the second stage.

We employ the EM algorithm for maximizing the above log-likelihood, since our model involves latent variables $\mathbf{s}_i$. Let $\mathcal{S}^j = (\mathbf{s}_1^j, \ldots, \mathbf{s}_{M_j}^j)$ denote the set of true sequences that generates $\bar{\mathcal{I}}^j$. The EM algorithm iteratively maximizes the following objective function, at iteration m

$$\begin{aligned} Q_j\left(\boldsymbol{\theta}_E, \boldsymbol{\theta}_E^{(m)}\right) &= \mathbb{E}\left[\ell\left(\boldsymbol{\theta}_{E,j} \;\middle|\; \bar{\mathcal{I}}^j, \mathcal{S}^j; \boldsymbol{\theta}_T^{(0)}\right) \;\middle|\; \bar{\mathcal{I}}^j, \boldsymbol{\theta}_{E,j}^{(m)}, \boldsymbol{\theta}_T^{(0)}\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{M_j} \ell_i\left(\boldsymbol{\theta}_{E,j} \;\middle|\; \boldsymbol{I}_i^j, \mathbf{s}_i^j; \boldsymbol{\theta}_T^{(0)}\right) \;\middle|\; \boldsymbol{I}_i^j, \boldsymbol{\theta}_{E,j}^{(m)}, \boldsymbol{\theta}_T^{(0)}\right], \end{aligned} \tag{3.11}$$

where $\ell_i\left(\boldsymbol{\theta}_{E,j} \;\middle|\; \boldsymbol{I}_i^j, \mathbf{s}_i^j; \boldsymbol{\theta}_T^{(0)}\right)$ is the complete data log-likelihood for the augmented complete data $(\boldsymbol{I}_i^j, \mathbf{s}_i^j)$ of cluster i.

Note that the estimation of $\boldsymbol{\theta}_{E,j}$ within tile j can be carried out independently of all other tiles. Henceforth, we will drop the subscript and superscript j from all related terms in order to simplify the notation.

For any given tile, the tile-wide complete-data log-likelihood factors into two components:

$$\begin{aligned}\ell_i\left(\boldsymbol{\theta}_E \;\middle|\; \boldsymbol{I}_i, \mathbf{s}_i; \boldsymbol{\theta}_T^{(0)}\right) &= \log f\left(\boldsymbol{I}_i, \mathbf{s}_i, \mid \boldsymbol{\theta}_E, \boldsymbol{\theta}_T^{(0)}\right) \\ &= \log f\left(\mathbf{s}_i | \boldsymbol{\theta}_T^{(0)}\right) + \log f\left(\boldsymbol{I}_i \mid \mathbf{s}_i; \boldsymbol{\theta}_E\right), \end{aligned} \tag{3.12}$$

which correspond to the emission and transition distributions respectively. Similarly, we can decompose the objective function as follows,

$$Q\left(\boldsymbol{\theta}_E, \boldsymbol{\theta}_E^{(m)}\right) = \sum_{i=1}^{M}\left\{\mathbb{E}\left[\log f\left(\mathbf{s}_i \mid \boldsymbol{\theta}_T^{(0)}\right) \;\middle|\; \boldsymbol{I}_i; \boldsymbol{\theta}_E^{(m)}, \boldsymbol{\theta}_T^{(0)}\right] + \mathbb{E}\left[\log f\left(\boldsymbol{I}_i \mid \mathbf{s}_i, \boldsymbol{\theta}_E\right) \;\middle|\; \boldsymbol{I}_i; \boldsymbol{\theta}_E^{(m)}, \boldsymbol{\theta}_T^{(0)}\right]\right\} \tag{3.13}$$

$$= \text{some constant that does not depend on } \boldsymbol{\theta}_E$$

$$+ \sum_{i=1}^{M}\left\{\sum_{\mathbf{s}_{i,k}}\left[\log g_1(\boldsymbol{I}_{i,1} \mid \mathbf{s}_{i,k}, \mathbf{x}_{i,k}) + \sum_{t=2}^{k}\log g_t(\boldsymbol{I}_{i,t} \mid \boldsymbol{I}_{i,t-1}, \mathbf{s}_{i,k}, \mathbf{x}_{i,k})\right]\boldsymbol{\phi}_{i,1}^{(m)}(\mathbf{s}_{i,k})\right\} \tag{3.14}$$

$$+ \sum_{i=1}^{M}\left[\sum_{t=k+1}^{l}\sum_{\mathbf{s}_{i,t}}\log g_t(\boldsymbol{I}_{i,t} \mid \boldsymbol{I}_{i,t-1}, \mathbf{s}_{i,t}, \mathbf{x}_{i,t})\boldsymbol{\phi}_{i,t}^{(m)}(\mathbf{s}_{i,t})\right], \tag{3.15}$$

where the $\boldsymbol{\phi}$'s are conditional pmfs defined as follows,

$$\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) \equiv P\left(\boldsymbol{S}_{i,t} = \boldsymbol{\omega} \mid \boldsymbol{I}_i; \boldsymbol{\theta}_E^{(m)}, \boldsymbol{\theta}_T^{(0)}\right), \; t \geq k. \tag{3.16}$$

In the context of the HMM, (3.16) can be computed efficiently using a dynamic programming algorithm, called the Forward-Backward algorithm [Rabiner (1989)]. The details of this algorithm are described in the next section.

3.2.3.3 Forward-backward algorithm (E-step)

Let us first consider the forward algorithm by defining the density function,

$$\boldsymbol{a}_{i,t}^{(m)}(\boldsymbol{\omega}) = f\left(\boldsymbol{I}_{i,1}, \ldots, \boldsymbol{I}_{i,t}, \mathbf{s}_{i,t^*} = \boldsymbol{\omega} \mid \boldsymbol{\theta}_E^{(m)}, \boldsymbol{\theta}_T^{(0)}\right), \quad t \leq k. \tag{3.17}$$

Then we have,

(i) Initialization:

$$\boldsymbol{a}_{i,k}^{(m)}(\boldsymbol{\omega}) = \mu(\widetilde{\boldsymbol{\omega}}) g_1(\boldsymbol{I}_{i,1} \mid \boldsymbol{\omega}, \mathbf{x}_{i,k}) \times \prod_{t=2}^{k} g_t(\boldsymbol{I}_{i,t} \mid \boldsymbol{I}_{i,t-1}, \boldsymbol{\omega}, \mathbf{x}_{i,k}). \tag{3.18}$$

(ii) Induction:

$$\boldsymbol{a}_{i,t+1}^{(m)}(\boldsymbol{\omega}) = \left[\sum_{\boldsymbol{\nu} \in \mathcal{N}_{it}(\mathbf{x}_{i,t}) : \overline{\boldsymbol{\nu}} \in \mathcal{T}(\overline{\boldsymbol{\omega}})} \boldsymbol{a}_{i,t}^{(m)}(\boldsymbol{\nu}) p(\boldsymbol{\omega} \mid \boldsymbol{\nu}) \right] g_{t+1}(\boldsymbol{I}_{i,t+1} \mid \boldsymbol{I}_{i,t}, \boldsymbol{\omega}, \mathbf{x}_{i,t+1}), \quad t \geq k. \tag{3.19}$$

Next, define the following function for the backward algorithm,

$$\boldsymbol{b}_{i,t}^{(m)}(\boldsymbol{\omega}) = f\left(\boldsymbol{I}_{i,t+1}, \ldots, \boldsymbol{I}_{i,l} \mid \mathbf{s}_{i,t^*} = \boldsymbol{\omega}; \boldsymbol{\theta}_E^{(m)}, \boldsymbol{\theta}_T^{(0)} \right). \tag{3.20}$$

Again, we can solve $\boldsymbol{b}_{i,t}^{(m)}(\boldsymbol{\omega})$ inductively, with:

(i) Initialization:

$$\boldsymbol{b}_{i,l}^{(m)}(\boldsymbol{\omega}) = 1, \quad \forall \boldsymbol{\omega} \in \mathcal{N}_{il}(\mathbf{x}_{i,l}), \quad t \leq k. \tag{3.21}$$

(ii) Induction:

a) For $k < t < l$,

$$\boldsymbol{b}_{i,t}^{(m)}(\boldsymbol{\omega}) = \left[\sum_{\boldsymbol{\nu} \in \mathcal{N}_{i,t+1}(\mathbf{x}_{i,t+1}) \cap \mathcal{T}(\boldsymbol{\omega})} p(\boldsymbol{\nu} \mid \boldsymbol{\omega}) g_{t+1}(\boldsymbol{I}_{i,t+1} \mid \boldsymbol{I}_{i,t}, \boldsymbol{\nu}, \mathbf{x}_{i,t+1}) \right] \boldsymbol{b}_{i,t+1}^{(m)}(\boldsymbol{\nu}), \tag{3.22}$$

b) And finally for $t = k$,

$$\boldsymbol{b}_{i,k}^{(m)}(\boldsymbol{\omega}) = g_1(\boldsymbol{I}_{i,1} \mid \boldsymbol{\omega}, \mathbf{x}_{i,k}) \times \prod_{t=2}^{k} g_t(\boldsymbol{I}_{i,t} \mid \boldsymbol{I}_{i,t-1}, \boldsymbol{\nu}, \mathbf{x}_{i,t}) \times \left[\sum_{\boldsymbol{\nu} \in \mathcal{N}_{i,k+1}(\mathbf{x}_{i,k+1}) \cap \mathcal{T}(\boldsymbol{\omega})} p(\boldsymbol{\nu} \mid \boldsymbol{\omega}) g_{k+1}(\boldsymbol{I}_{i,k+1} \mid \boldsymbol{I}_{i,k}, \boldsymbol{\nu}, \mathbf{x}_{i,k+1}) \right] \boldsymbol{b}_{i,k+1}^{(m)}(\boldsymbol{\omega}). \tag{3.23}$$

Using the inductively computed variables defined above, we can get the observed log-likelihood

$$f(\boldsymbol{I}_i) = \sum_{\boldsymbol{\omega} \in \mathcal{N}_l(\mathbf{x}_{i,l})} \boldsymbol{a}_{i,l}^{(m)}(\boldsymbol{\omega}), \tag{3.24}$$

and the conditional pdf

$$\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) = \frac{\boldsymbol{a}_{i,t}^{(m)}(\boldsymbol{\omega}) \boldsymbol{b}_{i,t}^{(m)}(\boldsymbol{\omega})}{\sum_{\boldsymbol{\nu} \in \mathcal{N}_{t^*}(\mathbf{x}_{i,t^*})} \boldsymbol{a}_{i,t}^{(m)}(\boldsymbol{\nu}) \boldsymbol{b}_{i,t}^{(m)}(\boldsymbol{\nu})}, \quad t \geq k. \tag{3.25}$$

3.2.3.4 M-step for transition parameters

The (penalized) estimation of transition parameters is independent from the estimation of emission parameters. The detailed discussion of the estimation procedure can be found in the Supplementary Information (§A.3.2.)

3.2.3.5 M-step for emission parameters

Updating α_t

Given the factorization of the Q function, we seek to update the cycle-dependent parameters $(\boldsymbol{C}_t, \alpha_t, \delta_t, \boldsymbol{\Sigma}_t)$ in the M-step, by maximizing (3.14) and/or (3.15). Let us define the following variables,

$$\Delta_t = \prod_{j=2}^{t}(1-\delta_t), t > 1; \quad \Delta_1 = 1. \tag{3.26}$$

$$\boldsymbol{u}_{i,t}(\boldsymbol{\omega}) = \boldsymbol{I}_{i,t} - \lambda_i \Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]} - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}, \quad t > 1. \tag{3.27}$$

Then, for $t > 1$, we have,

$$\begin{aligned}
&\frac{\partial Q(\boldsymbol{\theta}_E, \boldsymbol{\theta}_E^{(m)})}{\partial \alpha_t} \\
&= \frac{\partial}{\partial \alpha_t} \sum_{\boldsymbol{I}_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \Big[-\frac{4}{2}\log(2\pi) - \frac{1}{2}\log\left|\|\lambda_i \Delta_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\|_2^2 \boldsymbol{\Sigma}_t\right| \\
&\quad - \frac{1}{2}\|\lambda_i \Delta_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\|_2^{-2} \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \Big] \phi_{i,t}^{(m)}(\boldsymbol{\omega}).
\end{aligned} \tag{3.28}$$

Note that $\boldsymbol{\Sigma}_t$ and hence $\boldsymbol{\Sigma}_t^{-1}$ are symmetric, which leads to,

$$\frac{\partial \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega})}{\partial \alpha_t} = \frac{\partial \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega})}{\partial \boldsymbol{u}_{i,t}(\boldsymbol{\omega})} \frac{\partial \boldsymbol{u}_{i,t}(\boldsymbol{\omega})}{\partial \alpha_t} = 2\boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1}[(1-\delta_t)\boldsymbol{I}_{i,t-1}].$$

Plugging in the above partial derivative back into (3.28), we have,

$$\sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} (\lambda_i \Delta_t)^{-2} \left[\boldsymbol{I}_{i,t} - \lambda_i \Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]} - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}\right]' \boldsymbol{\Sigma}_t^{-1}[(1-\delta_t)\boldsymbol{I}_{i,t-1}] \phi_{i,t}^{(m)}(\boldsymbol{\omega}) = 0.$$

Cancelling $(1-\delta_t)$ and rearranging yields

$$\begin{aligned}
&\sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \alpha_t (1-\delta_t)(\lambda_i \Delta_t)^{-2} \boldsymbol{I}_{i,t-1}' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{I}_{i,t-1} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \phi_{i,t}^{(m)}(\boldsymbol{\omega}) \\
= \quad &\sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} (\lambda_i \Delta_t)^{-2} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \left[\boldsymbol{I}_{i,t} - \lambda_i \Delta_t \left(\boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)\right]' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{I}_{i,t-1} \phi_{i,t}^{(m)}(\boldsymbol{\omega}).
\end{aligned}$$

We know from (3.25) that

$$\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) = 1, \forall i, t. \tag{3.29}$$

Therefore, from the above equation we derive the formula to update α_t,

$$\alpha_t^{(m+1)} = \frac{\sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}(\lambda_i\Delta_t)^{-2}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})}\left[\boldsymbol{I}_{i,t} - \lambda_i\Delta_t\left(\boldsymbol{C}_t\boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)\right]'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{I}_{i,t-1}\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{(1-\delta_t)\sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}(\lambda_i\Delta_t)^{-2}\boldsymbol{I}_{i,t-1}'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{I}_{i,t-1}}. \tag{3.30}$$

Updating $\boldsymbol{C}_t$

To update $\boldsymbol{C}_t$, notice that for $\boldsymbol{C}_t = \begin{bmatrix}\boldsymbol{c}_t^A & \boldsymbol{c}_t^C & \boldsymbol{c}_t^G & \boldsymbol{c}_t^T\end{bmatrix}$, $\boldsymbol{C}_t\boldsymbol{e}_b = \boldsymbol{c}_t^b$ corresponds to the column indexed by the nucleotide $b \in \{A, C, G, T\}$. For a given column $\boldsymbol{c}_t^b$, it can be shown that,

$$\begin{aligned}
&\frac{\partial Q\left(\boldsymbol{\theta}_E, \boldsymbol{\theta}_E^{(m)}\right)}{\partial \boldsymbol{c}_t^b} \\
&= \sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}\frac{\partial}{\partial \boldsymbol{c}_t^b}\left[-\frac{1}{2}(\lambda_i\Delta_t)^{-2}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})\right] \\
&= -\sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}\left[(\lambda_i\Delta_t)^{-2}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\frac{\partial \boldsymbol{u}_{i,t}(\boldsymbol{\omega})}{\partial \boldsymbol{c}_t^b}\right]\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}).
\end{aligned}$$

Using $\partial \boldsymbol{u}_{i,t}(\boldsymbol{\omega})/\partial \boldsymbol{c}_t^b = -\lambda\Delta_t$, we have

$$\begin{aligned}
&\frac{\partial Q\left(\boldsymbol{\theta}_E, \boldsymbol{\theta}_E^{(m)}\right)}{\partial \boldsymbol{c}_t^b} = 0 \\
\Longrightarrow \quad &\sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}\left[(\lambda_i\Delta_t)^{-1}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\right]\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) = 0 \\
\Longrightarrow \quad &\sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}\boldsymbol{c}_t^{b\prime}\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) = \\
&\qquad \sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}(\lambda_i\Delta_t)^{-1}\left[\left(\boldsymbol{I}_{i,t}' - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}'\right)\boldsymbol{\Sigma}_t^{-1}\right]\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) \\
\Longrightarrow \quad &\sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}\boldsymbol{c}_t^{b\prime}\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) = \\
&\qquad \sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}(\lambda_i\Delta_t)^{-1}\left[\boldsymbol{I}_{i,t}' - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}'\right]\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})
\end{aligned}$$

Then it is straightforward to show that the formula to update $\boldsymbol{c}_t^b$ is

$$\boldsymbol{c}_t^{b(m+1)} = \frac{\sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}(\lambda_i\Delta_t)^{-1}\left[\boldsymbol{I}_{i,t}' - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}'\right]\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{\sum_{\boldsymbol{I}_i\in\bar{\mathcal{I}}}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}[k] = b\}\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}. \tag{3.31}$$

To avoid identifiability issues, we need to renormalize the updated cross-talk matrix $\boldsymbol{C}_t^{(m+1)}$. Li and Speed (1999) suggested to normalize $\boldsymbol{C}_t^{(m+1)}$ such that each column sums up to 1. In practice, however, the relative intensities are different across the four channels. To properly model this effect, we renormalize the cross-talk matrix so that the largest column sums to 1. All the other columns are scaled accordingly.

Updating $\boldsymbol{\Sigma}_t$

Using matrix calculus, we can show that,

$$\frac{\partial Q(\boldsymbol{\theta}_E, \boldsymbol{\theta}_E^{(m)})}{\partial \boldsymbol{\Sigma}_t} = \sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \frac{\partial}{\partial \boldsymbol{\Sigma}_t} \left[-\frac{1}{2} \log \left| \boldsymbol{\Sigma}_t \right| - \frac{1}{2} (\lambda_i \Delta_t)^{-2} tr \left[\boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \right] \right] \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}), \tag{3.32}$$

where

$$\frac{\partial \; tr \left[\boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \right]}{\partial \boldsymbol{\Sigma}_t} = \frac{\partial \; tr \left[\boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \right]}{\partial \boldsymbol{\Sigma}_t} = -\boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1}, \tag{3.33}$$

$$\frac{\partial \log |\boldsymbol{\Sigma}_t|}{\partial \boldsymbol{\Sigma}_t} = \frac{1}{|\boldsymbol{\Sigma}_t|} |\boldsymbol{\Sigma}_t| \boldsymbol{\Sigma}_t^{-1} = \boldsymbol{\Sigma}_t^{-1}. \tag{3.34}$$

Then,

$$\begin{aligned}
& \frac{\partial Q(\boldsymbol{\theta}_E, \boldsymbol{\theta}_E^{(m)})}{\partial \boldsymbol{\Sigma}_t} = \mathbf{0} \\
\implies \quad & \sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \left[-\frac{1}{2} \boldsymbol{\Sigma}_t^{-1} + \frac{1}{2} (\lambda_i \Delta_t)^{-2} \boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} \right] \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) \\
\implies \quad & \boldsymbol{\Sigma}_t^{-1} \sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) = \sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} (\lambda_i \Delta_t)^{-2} \boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) \\
\implies \quad & \boldsymbol{\Sigma}_t \sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) = \sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} (\lambda_i \Delta_t)^{-2} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}).
\end{aligned}$$

It easily follows, using the results from (3.29), that

$$\begin{aligned}
\boldsymbol{\Sigma}_t^{(m+1)} &= \frac{\sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} (\lambda_i \Delta_t)^{-2} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{\sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})} \\
&= \frac{\sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} (\lambda_i \Delta_t)^{-2} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{M}.
\end{aligned} \tag{3.35}$$

Updating δ_t

Since there is no closed-form formula to update δ_t, we used an approximation, which is similar to the one proposed in Kao et al. (2009) (Supplementary data). Specifically, we used a slightly modified model of Eq. (3.8) for the emission of $\boldsymbol{I}_{i,t}, t > 1$, that is,

$$\boldsymbol{I}_{i,t}|\boldsymbol{I}_{i,t-1}, \mathbf{s}_{i,t^*}, \mathbf{x}_{i,t^*} \sim \mathbf{1}\{\mathbf{s}_{i,t^*} \in \mathcal{N}_{i,t^*}(\mathbf{x}_{i,t^*})\} \times N\left(\Lambda^*_{i,t}\boldsymbol{C}_t\boldsymbol{e}_{\mathbf{s}_i[t]} + \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}, \|\Lambda^{(m)}_{i,t}\boldsymbol{e}_{\mathbf{s}_i[t]}\|_2^2\boldsymbol{\Sigma}_t\right), \tag{3.36}$$

where

$$\Lambda^*_{i,t} = \lambda_i\left[\prod_{j=2}^{t-1}(1-\delta_j^{(m)})\right](1-\delta_t), \tag{3.37}$$

$$\Lambda^{(m)}_{i,t} = \lambda_i\prod_{j=2}^{t}(1-\delta_j^{(m)}). \tag{3.38}$$

Adopting the two modifications (3.37) and (3.38) make the partial derivative of (3.36) with respect to δ_t linear in δ_t, so

$$\begin{aligned}
&\frac{\partial Q\left(\boldsymbol{\theta}_E, \boldsymbol{\theta}_E^{(m)}\right)}{\partial\delta_t} = 0 \\
\implies \quad &\sum_{\boldsymbol{I}_i}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t^*})}\frac{\partial}{\partial\delta_t}\left[-\frac{1}{2}\left[\lambda_i\Delta_t^{(m)}\right]^{-2}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})\boldsymbol{\phi}^{(m)}_{i,t}(\boldsymbol{\omega})\right] = 0 \\
\implies \quad &\sum_{\boldsymbol{I}_i}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t^*})}\left[\left[\lambda_i\Delta_t^{(m)}\right]^{-2}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\frac{\partial\boldsymbol{u}_{i,t}(\boldsymbol{\omega})}{\partial\delta_t}\boldsymbol{\phi}^{(m)}_{i,t}(\boldsymbol{\omega})\right] = 0.
\end{aligned} \tag{3.39}$$

For simplicity, define,

$$\boldsymbol{v}_{i,t}(\boldsymbol{\omega}) \equiv -\frac{\partial\boldsymbol{u}_{i,t}(\boldsymbol{\omega})}{\partial\delta_t} = \lambda_i\Delta_{t-1}\boldsymbol{C}_t\boldsymbol{e}_{\boldsymbol{\omega}[k]} + \alpha_t\boldsymbol{I}_{i,t-1}. \tag{3.40}$$

Then,

$$\begin{aligned}
&\sum_{\boldsymbol{I}_i}\left(\lambda_i\Delta_t^{(m)}\right)^{-2}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t^*})}\left[\lambda_i\Delta_{t-1}\left(1-\delta_t^{(m)}\right)\right]^{-2}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{v}_{i,t}(\boldsymbol{\omega}) = 0 \\
\implies \quad &\sum_{\boldsymbol{I}_i}\left(\lambda_i\Delta_t^{(m)}\right)^{-2}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t^*})}\boldsymbol{I}'_{i,t}\boldsymbol{\Sigma}_t^{-1}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})\boldsymbol{\phi}^{(m)}_{i,t}(\boldsymbol{\omega}) = \\
&\qquad (1-\delta_t)\sum_{\boldsymbol{I}_i}\left(\lambda_i\Delta_t^{(m)}\right)^{-2}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t^*})}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})\boldsymbol{\phi}^{(m)}_{i,t}(\boldsymbol{\omega}).
\end{aligned} \tag{3.41}$$

From (3.41), we have,

$$\delta_t^{(m+1)} = 1 - \frac{\sum_{\boldsymbol{I}_i}\left(\lambda_i\Delta_t^{(m)}\right)^{-2}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t^*})}\boldsymbol{I}'_{i,t}\boldsymbol{\Sigma}_t^{-1}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})\boldsymbol{\phi}^{(m)}_{i,t}(\boldsymbol{\omega})}{\sum_{\boldsymbol{I}_i}\left(\lambda_i\Delta_t^{(m)}\right)^{-2}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{it}(\mathbf{x}_{i,t^*})}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})\boldsymbol{\phi}^{(m)}_{i,t}(\boldsymbol{\omega})}. \tag{3.42}$$

We did not drop the constant $\Delta_t^{(m)}$ from both the numerator and the denominator, for the ease of derivation in upcoming §3.2.3.6.

Updating λ_i

In order to update λ_i in the M-step, we seek to maximize the following objective function:

$$Q_i\left(\lambda_i, \boldsymbol{\theta}_E^{(m)}\right) = \sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{t^*}(\mathbf{x}_{i,t^*})} \left[-\frac{1}{2}\log|(\lambda_i\Delta_t)^2\boldsymbol{\Sigma}_t| - \frac{1}{2}(\lambda_i\Delta_t)^{-2}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \right] \phi_{i,t}^{(m)}(\boldsymbol{\omega}) - \eta(\lambda_i - \overline{\Lambda^j})^2. \tag{3.43}$$

The ℓ_2-penalty term $\eta(\lambda_i - \overline{\Lambda^j})^2$ is introduced to avoid over-fitting by regressing the cluster-specific estimate $\hat{\lambda}_i$ toward the tilde-wide mean intensity $\overline{\Lambda^j}$.

Like δ_t, it is not trivial to find the solution to the equation $\partial Q_i\left(\lambda_i, \boldsymbol{\theta}_E^{(m)}\right)/\partial\lambda_i = 0$. However, since λ_i is a cluster-specific parameter, we can employ numerical methods to maximize (3.43). First, we can show that $\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})$ expands to,

$$\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{u}_{i,t}(\boldsymbol{\omega}) = \left[(\boldsymbol{I}_{i,t} - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}) - \lambda_i\Delta_t\boldsymbol{C}_t\boldsymbol{e}_{\boldsymbol{\omega}[k]}\right]' \boldsymbol{\Sigma}_t^{-1} \left[(\boldsymbol{I}_{i,t} - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}) - \lambda_i\Delta_t\boldsymbol{C}_t\boldsymbol{e}_{\boldsymbol{\omega}[k]}\right]. \tag{3.44}$$

Letting $\boldsymbol{\epsilon}_{i,t} \equiv \boldsymbol{I}_{i,t} - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}$ in (3.44), then we have,

$$\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{u}_{i,t}(\boldsymbol{\omega}) = \boldsymbol{\epsilon}_{i,t}'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\epsilon}_{i,t} + (\lambda_i\Delta_t)^2\left(\boldsymbol{C}_t\boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)'\boldsymbol{\Sigma}_t^{-1}\left(\boldsymbol{C}_t\boldsymbol{e}_{\boldsymbol{\omega}[k]}\right) - 2(\lambda_i\Delta_t)\left(\boldsymbol{C}_t\boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)'\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\epsilon}_{i,t}. \tag{3.45}$$

Using the results of (3.45), (3.43) can be expanded to:

$$
\begin{aligned}
Q_i\left(\lambda_i, \boldsymbol{\theta}_E^{(m)}\right) &= c_1 - \eta(\lambda_i - \overline{\Lambda^j})^2 + \sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{i,t^*}(\mathbf{x}_{i,t^*})} -4\log(\lambda_i)\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) \\
&\quad - \sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{i,t^*}(\mathbf{x}_{i,t^*})} \frac{1}{2}(\lambda_i \Delta_t)^{-2} \Big[\boldsymbol{\epsilon}_{i,t}' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\epsilon}_{i,t} + (\lambda_i \Delta_t)^2 \left(\boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)' \boldsymbol{\Sigma}_t^{-1} \left(\boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right) \\
&\quad - 2(\lambda_i \Delta_t) \left(\boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\epsilon}_{i,t} \Big] \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) \\
&= c_2 + l \log \lambda_i - \frac{1}{2}\lambda^{-2} \left[\sum_{t=1}^{l} \Delta_t^{-2} \boldsymbol{\epsilon}_{i,t}' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\epsilon}_{i,t} \right] \\
&\quad + \lambda_i^{-1} \left[\sum_{t=1}^{l} \Delta_t^{-1} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{i,t^*}(\mathbf{x}_{i,t^*})} (\boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]})' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\epsilon}_{i,t} \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) \right] - \eta(\lambda_i - \overline{\Lambda^j})^2, \qquad (3.46)
\end{aligned}
$$

where c_1 and c_2 are some constants that do not depend on λ_i. Notice that both summations in (3.46) do not involve λ_i. This implies that (3.46) can be evaluated very efficiently as a function of λ_i, justifying the usage of numerical optimization.

Alternatively, we can consider the same approximation applied to δ_t for λ_i, so that the numerical optimization can be avoided in order to speed up the computation. To do so, consider the following modified emission distribution,

$$
\boldsymbol{I}_{i,t} | \boldsymbol{I}_{i,t-1}, \mathbf{s}_{i,t^*}, \mathbf{x}_{i,t^*} \sim \mathbf{1}\{\mathbf{s}_{i,t^*} \in \mathcal{N}_{i,t^*}(\mathbf{x}_{i,t^*})\} \times N\left(\Lambda_{i,t} \boldsymbol{C}_t \boldsymbol{e}_{\mathbf{s}_i[t]} + \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}, \|\widehat{\Lambda_{i,t}} \boldsymbol{e}_{\mathbf{s}_i[t]}\|_2^2 \boldsymbol{\Sigma}_t\right),
$$

where

$$
\widehat{\Lambda_{i,t}} = \lambda_i^{(m)} \Delta_t. \qquad (3.47)
$$

Accordingly, we can update (3.43), applying above approximation, and get,

$$
\begin{aligned}
&Q_i\left(\lambda_i, \boldsymbol{\theta}_E^{(m)}\right) = \\
&\quad \sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{t^*}(\mathbf{x}_{i,t^*})} \left[-\frac{1}{2} \log \left| (\lambda_i^{(m)} \Delta_t)^2 \boldsymbol{\Sigma}_t \right| - \frac{1}{2} \left(\lambda_i^{(m)} \Delta_t\right)^{-2} \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} \boldsymbol{u}_{i,t}(\boldsymbol{\omega}) \right] \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) - \eta(\lambda_i - \overline{\Lambda^j})^2.
\end{aligned} \qquad (3.48)
$$

It is then straightforward to show that,

$$\begin{aligned}
&\frac{\partial Q_i\left(\lambda_i, \boldsymbol{\theta}_E^{(m)}\right)}{\partial \lambda_i} = 0 \\
\Longrightarrow \quad & \sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{t^*}(\mathbf{x}_{i,t^*})} \boldsymbol{u}_{i,t}(\boldsymbol{\omega})' \boldsymbol{\Sigma}_t^{-1} (\Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]} \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) - 2\eta(\lambda_i - \overline{\Lambda^j}) = 0 \\
\Longrightarrow \quad & \lambda_i \left[\sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{t^*}(\mathbf{x}_{i,t^*})} \left(\Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)' \boldsymbol{\Sigma}_t^{-1} \left(\Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right) \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}) + 2\eta \right] = \\
& \qquad 2\eta \overline{\Lambda^j} + \sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{t^*}(\mathbf{x}_{i,t^*})} \left[\boldsymbol{I}_{i,t} - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}\right]' \boldsymbol{\Sigma}_t^{-1} \left(\Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right) \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega}).
\end{aligned}$$

Thus, the (regularized) update formula for λ_i is given by,

$$\lambda_i^{(m+1)} = \frac{2\eta\overline{\Lambda^j} + \sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{t^*}(\mathbf{x}_{i,t^*})} \left[\boldsymbol{I}_{i,t} - \alpha_t(1-\delta_t)\boldsymbol{I}_{i,t-1}\right]' \boldsymbol{\Sigma}_t^{-1} \left(\Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right) \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{2\eta + \sum_{t=1}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{t^*}(\mathbf{x}_{i,t^*})} \left(\Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)' \boldsymbol{\Sigma}_t^{-1} \left(\Delta_t \boldsymbol{C}_t \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right) \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}. \tag{3.49}$$

Obviously, as $\eta \to \infty$, $\lambda_i^{(m+1)} \to \overline{\Lambda^j}$.

3.2.3.6 Cycle-dependency

To avoid over-fitting the cycle-dependent parameters $(\boldsymbol{C}_t, \alpha_t, \delta_t, \boldsymbol{\Sigma}_t)$, a common approach is to bin adjacent cycles into windows [Kao et al. (2009); Das and Vikalo (2013)]. That is, for a given window width W, we divide the l cycles into $l_W = \lceil \frac{l}{W} \rceil$ windows, indexed by $w = 1, 2, \ldots, l_W$. For each given window w, its parameters $(\boldsymbol{C}_w, \alpha_w, \delta_w, \boldsymbol{\Sigma}_w)$ can be estimated simply by combining the information from cycles $t \in T_w$, where T_w is defined by,

$$T_w = [(w-1) * W + 1, \ \min(w * W, l)].$$

Starting from the cycle-dependent formulas, (3.30), (3.31), (3.35) and (3.42), we can directly derive that,

$$\alpha_w^{(m+1)} = \frac{\sum_{i=1}^{M} \left(\lambda_i \bar{\Delta}_t\right)^{-2} \sum_{t \in T_w} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \left[\boldsymbol{I}_{i,t} - \lambda_i \bar{\Delta}_t \left(\boldsymbol{C}_w \boldsymbol{e}_{\boldsymbol{\omega}[k]}\right)\right]' \boldsymbol{\Sigma}_w^{-1} \boldsymbol{I}_{i,t-1} \boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{(1-\delta_w) \sum_{\boldsymbol{I}_i \in \bar{\mathcal{I}}} \sum_{t \in T_w} \left(\lambda_i \bar{\Delta}_t\right)^{-2} \boldsymbol{I}'_{i,t-1} \boldsymbol{\Sigma}_w^{-1} \boldsymbol{I}_{i,t-1}}, \tag{3.50}$$

where $\bar{\Delta}_t$ is defined as

$$\bar{\Delta}_t = \prod_{j=2}^{t} \left(1 - \delta_{\lceil \frac{t}{W} \rceil}\right), \ t > 1; \qquad \bar{\Delta}_t = 1,$$

and that

$$\boldsymbol{c}_w^{b\ (m+1)} = \frac{\sum\limits_{i,t,\boldsymbol{\omega}} \mathbb{1}\{\boldsymbol{\omega}[k]=b\}(\lambda_i\Delta_t)^{-1}\left[\boldsymbol{I}'_{i,t} - \alpha_w(1-\delta_t)\boldsymbol{I}'_{i,t-1}\right]\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{\sum\limits_{i,t,\boldsymbol{\omega}} \mathbb{1}\{\boldsymbol{\omega}[k]=b\}\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}, \tag{3.51}$$

$$\boldsymbol{\Sigma}_w^{(m+1)} = \frac{\sum\limits_{i,t,\boldsymbol{\omega}} (\lambda\bar{\Delta}_t)^{-2}\boldsymbol{u}_{i,t}(\boldsymbol{\omega})\boldsymbol{u}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{MW}, \tag{3.52}$$

$$\delta_w^{(m+1)} = 1 - \frac{\sum\limits_{i,t,\boldsymbol{\omega}} \left(\lambda_i\bar{\Delta}_t^{(m)}\right)^{-2}\boldsymbol{I}'_{i,t}\boldsymbol{\Sigma}_w^{-1}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}{\sum\limits_{i,t,\boldsymbol{\omega}} \left(\lambda_i\bar{\Delta}_t^{(m)}\right)^{-2}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})'\boldsymbol{\Sigma}_w^{-1}\boldsymbol{v}_{i,t}(\boldsymbol{\omega})\boldsymbol{\phi}_{i,t}^{(m)}(\boldsymbol{\omega})}, \tag{3.53}$$

with $\sum\limits_{i,t,\boldsymbol{\omega}}$ as a short hand for $\sum_{i=1}^{M}\sum_{t\in T_w}\sum_{\boldsymbol{\omega}\in\mathcal{N}_{i,t^*}(\mathbf{x}_{i,t^*})}$.

For $\delta_w^{(m+1)}$, we need a slightly modified $\boldsymbol{v}_{i,t}(\boldsymbol{\omega})$, which is,

$$\boldsymbol{v}_{i,t}(\boldsymbol{\omega}) = \lambda_i\Delta_{t-1}\boldsymbol{C}_w\boldsymbol{e}_{\boldsymbol{\omega}[k]} + \alpha_w\boldsymbol{I}_{i,t-1}. \tag{3.54}$$

3.2.4 Base-calling and quality scores

Similar to [Das and Vikalo (2013)], once the model parameters are fitted by the expectation-maximization algorithm, the base-calls of $\boldsymbol{I}_i$ can be produced by either the Viterbi algorithm, or the forward-backward algorithm. The former identifies the most likely state sequence $\operatorname{argmax} P(\mathbf{s}_i \mid \boldsymbol{I}_i, \widehat{\boldsymbol{\theta}})$, where $\widehat{\boldsymbol{\theta}}$ is the estimated parameter vector. Since the Viterbi algorithm does not assess the probability of base-calling error on each individual base, another round of forward-backward algorithm is employed to compute the posterior probabilities

$$P\left(\boldsymbol{S}_{i[t]} = b \mid \boldsymbol{I}_i; \widehat{\boldsymbol{\theta}}\right) = \sum_{\boldsymbol{\omega}\in\mathcal{N}_{i,t^*}(\mathbf{x}_{i,t^*})} P\left(\boldsymbol{S}_{i,t} = \boldsymbol{\omega} \mid \boldsymbol{I}_i; \widehat{\boldsymbol{\theta}}\right)\mathbb{1}\{\boldsymbol{\omega}[\min\{t,k\}] = b\},\ b\in\Omega=\{A,C,G,T\}. \tag{3.55}$$

Each posterior probability p is in turn discretized and converted into quality score q using the formula $q = \lfloor -10\log_{10}(p)\rfloor$. Alternatively, the nucleotide that maximizes the posterior probability (3.55) can be directly treated as the base-call. To match convention, we cap the maximum quality score at $q = 40$. For our performance analysis in §3.3, we use the forward-backward algorithm to generate base-calls.

3.3 Results

As an integrated base-caller and error-corrector, we seek to evaluate PREMIER-bc's merits on both tasks. First, to assess the base-calling aspect of PREMIER-bc, we evaluate its performance against the following publicly available base-calling software: AYB [Massingham and Goldman (2012)], Softy [Das and Vikalo (2013)] and freeIbis [Renaud et al. (2013)], which are among the top-performers according to a recent review [Cacho et al. (2015)]. Next, we study how much unified approach improves error correction by comparing PREMIER-bc to its error correction counterpart PREMIER (using the reads called by Bustard.)

For our comparison, we employed the bacteriophage ϕ-X174 dataset used in [Cacho et al. (2015)], generated by the Illumina Genome Analyzer II sequencer. The dataset consists of a full lance of 100 tiles, each containing between 70,000 and 80,000 reads of length 76bp. We ran the model-based software, AYB and Softy, with their default settings on all tiles. Softy provides two options for base-calling: with the forward-backward algorithm (Softy-FB), or the soft output Viterbi algorithm (Softy-SOVA.) We ran both versions in our experiments. For freeIbis, which requires control samples for model training, the intensities and base-calls for the first tile were utilized for supervised learning. For comparing PREMIER and PREMIER-bcon an equal footing, both software were supplied with the following common model parameters: $k = 16, d = 8, \rho = 1,$ and $\gamma = 10^{-20}$. For the ℓ_2 penalty in PREMIER-bc, we set $\eta = 1000$. After base-calling, we conducted necessary post-processing of Softy output, in order to convert the their continuous probabilities of base-call quality into discrete quality scores.

To assess the base-calling quality of each software, the base-called reads were aligned to the known reference genome of ϕ-X174 (NC_001422.1), using the short read aligner BWA (v0.7.10) [Li and Durbin (2009)]. We used the default alignment parameters of the BWA-MEM algorithm, except for setting the read terminal clipping penalty to 100 (via option `-L 100,100`) to suppress the clipping of low-quality 5' or 3' regions of reads, so that the number of errors within those regions can be measured more precisely. From the alignment results, we computed the alignment rate and error rate for each tile.

Table 3.1: The comparison of base-calling quality in terms of alignment rate and error rate.

Base-caller	**Alignment rate**	**Mismatches**	**Error rate** ($\times 10^{-2}$)
AYB	94.73%	40852.3	0.767
Bustard	93.98%	53724.8	1.026
freeIbis	94.55%	44989.2	0.846
PREMIER-bc	**96.38%**	**27185.7**	**0.502**
Softy-FB	94.11%	44041.5	0.834
Softy-SOVA	94.11%	44574.2	0.842
PREMIER	96.21%	29440.4	0.545

Table 3.1 compares the base-calling quality in terms of average alignment rate and error rate across all tiles. Compared with the reference method, Bustard, all other base-callers produced reads with higher alignment rates, and lower error rates, signaling improved base-calling quality. Among all software, PREMIER-bc ranks atop, outperforming the runner-up, AYB, with sizeable margins. Specifically, PREMIER-bcimproved the alignment rate by 1.65%, while reducing the number of sequencing errors by 34.6%.

When compared with the sequential application of base-calling (Bustard) and error correction (PREMIER), PREMIER-bc managed to beat the pipelining approach with modest improvements in all metrics, indicative of the advantage of directly modeling the intensity data.

For more comprehensive comparisons, Figure 3.1 and 3.2 compare the software in terms of per-cycle error rate and per-tile error rate. In Figure 3.1, both PREMIER and PREMIER-bc; produced tile-sequences with significantly lower error rates than other base-callers. We note that the two outliers for PREMIER-bc correspond to failed model estimation, possibly due to the random subsampling of clusters. In most other cases, PREMIER-bc managed to produce data with less errors than its error-correction counterpart. Interestingly, Figure 3.2 provides an explanation for where PREMIER-bc performed better than PREMIER. Since both PREMIER and PREMIER-bc uses $k = 16$, the steep plummet of error rates at cycle 17 for these sibling software clearly reproduces a previously known weakness of our probabilistic framework: the limited error correction capacity within the first $k$mer. The reason is twofold. First, the first $k$mer cannot utilize upstream context (*i.e.* transitions) to identify errors. Secondly, to

reduce model complexity, the Hamming distance for the first $k$mer neighborhood, d_{it}, is often smaller than d_{it} for $t > k$. Still, within the first few cycles, PREMIER-bc produces base-calls with comparable error rates to freeIbis, which has the lowest error rates among traditional base-callers, and significantly improves the performance over that of PREMIER. This contrast shows that the raw intensities are highly informative in yielding accurate base-calls, when the genome content that error correctors rely on is less instructive. In later cycles, the error rates of both PREMIER and PREMIER-bc; creep upward. However, the rate of increment is modest at best, in comparison with other base-calling methods, demonstrating the usefulness of utilizing sequence dependence in the genome, when the noise level in the observed intensities are undesirable for reliable base-calling.

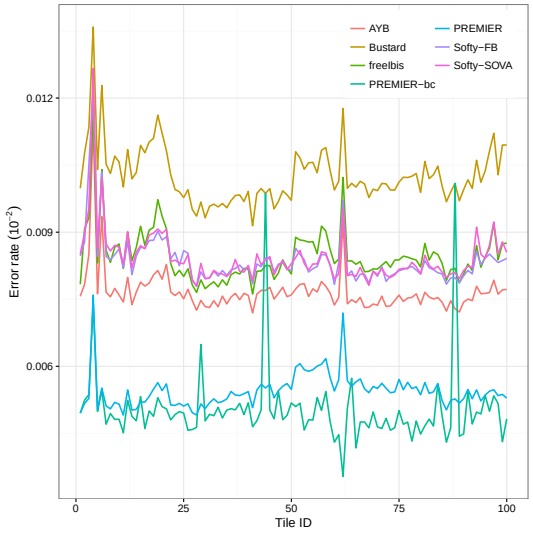


Figure 3.1: Comparison of per-tile error rate on the ϕ-X174 dataset

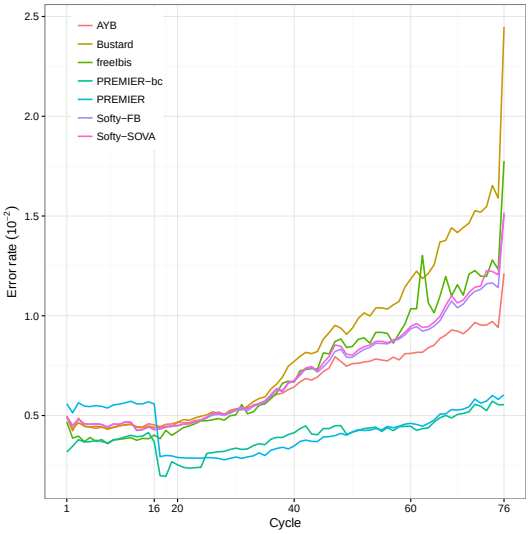


Figure 3.2: Comparison of per-cycle error rate on the ϕ-X174 dataset

3.4 Discussion

Base-calling and error correction have long been considered two different problems in bioinformatics. Base-callers are highly specialized in analyzing sequencer-specific signals and error characteristics, whereas error correctors operate on nucleotide sequences that are platform-independent, and exploit the underlying genome for identifying errors. The vast difference between input data format also leads to distinct problem-solving approaches. The Illumina

base-callers, for instance, are often model-based, utilizing statistical models to explicit model the sources of sequencing errors, and to probabilistically assess the likelihoods of nucleotides at a given cycle. Error correction methods, on the other hand, are mostly algorithmic and heuristic, exploiting data structures and algorithms to identify sequence exact/inexact matches or overlaps.

In this work, we consider an alternative approach to unify base-calling and error correction by recognizing that the sequencing process of the Illumina sequencer can be well modeled by a Hidden Markov model. As a generative model, the HMM consists of a hidden Markov process, which we utilized to model the underlying sequencing-by-synthesis (SBS) process. In particular, the newly synthesized bases are not random, rather they are complementary to the single stranded DNA template derived from a genome with limited length, so the next base can often be reliably predicted given the context. Conditional on the synthesized base, the HMM emits the observed intensity values, with an emission distribution that accounts for the overlap of fluorescence spectra, DNA degradation along cycles, and signal leakage between cycles. Our implementation, PREMIER-bc, is merely a simple form of a more general probabilistic framework that can potentially adapt to any base-calling and error-correction problems of sequencing platforms that utilize SBS principles.

In our base-calling experiments, we demonstrate that PREMIER-bc outperforms all existing Illumina base-callers with significant margins. This enhancement in base-calling quality by PREMIER-bc underscores the importance of simultaneously modeling the underlying genome sequence, and the intensity data. Breaking down the base-calling performance by cycle reveals that the raw intensities significantly improve accuracy within the first $k$mer, a known weakness of PREMIER. Conversely, toward the 3' end, the genome information is crucial in reducing error rate.

These observations invite rethinking of bioinformatics pipeline designs. Quite often, bioinformatics tools have the tendency to compress data by committing to deterministic outputs. The compression itself is rarely lossless, and thus reduces the amount of information that can be available downstream in the pipeline. Bustard prefers, notoriously, to assign special quality scores (Phred quality score 2) as indicators of low-quality regions, which unfortunately censors

the base uncertainties that are critical in resolving ambiguities in the genome. Therefore, it is reasonable to expect the direct modeling of raw intensities to improve the base qualities in read segments with non-informative quality scores, for instance in low quality reads, or approaching the 3' ends of reads.

Moreover, imagine an extension to our probabilistic model that directly aligns the raw intensities to a known reference. This is attainable by directly constructing the HMM state space from the genome sequence, instead of from the observed read sequences. This extended model can be utilized to compute the most likely alignment that maximizes the posterior probability $P(\boldsymbol{S}_i \mid \boldsymbol{I}_i, \boldsymbol{\theta})$, given the observed intensity matrix $\boldsymbol{I}_i$. And there's more to that. Now that all $k$mers in the state space are genomic $k$mers from the reference, differences between read sequence $\mathbf{x}_i$ and state sequence $\mathbf{s}_i$ may be attributed to either sequencing error, or true genomic variants. To predict variant versus error, one possibility is to utilize a finite mixture distribution as the emission density. The first component, which models noise, can assume similar form to the emission density of PREMIER-bc, whereas the second component explicitly models variants (perhaps based on the coverage of the observed $k$mer). Clearly, the intensity data provides more subtlety to probabilistically assess the likelihood of noise at a certain cycle, and thus modeling $\boldsymbol{I}_i$ has more statistical power than using $(\mathbf{x}_i, \mathbf{y}_i)$ alone, especially when $\mathbf{y}_i$ is less informative (e.g. censored by Bustard). We expect this model to outperform existing algorithm-based aligners, especially for noisy reads and in repetitive genomic regions.

Despite the optimistic results, we note that sequencing redundancy is key to attain good performance with PREMIER-bc. Therefore, PREMIER-bc is not suited to base-call individual tiles one at a time, especially for the sequencing experiments of large genomes. We plan to test PREMIER-bc on more datasets with varying sequencing coverage and genome complexity to extensively study the performance and robustness of our unified model.

CHAPTER 4. PREMIER-INDEL: PROBABILISTIC FRAMEWORK FOR CORRECTING INSERTION AND DELETION ERRORS IN HIGH-THROUGHPUT SEQUENCING READS

Abstract

Error correction of noisy reads obtained from high-throughput DNA sequencers is an important problem since read quality significantly affects downstream analyses such as detection of genetic variation and the complexity and success of sequence assembly. Most of the current error correction algorithms are only capable of recovering substitution errors. In this work, we present PREMIER-indel, an algorithm that simultaneously corrects insertion, deletion and substitution errors in reads from next generation DNA sequencing platforms. PREMIER-indel corrects insertion, deletion and substitution errors by modelling the sequencer output as emissions of an appropriately defined Hidden Markov Model (HMM). Reads are corrected to the corresponding maximum likelihood paths using an appropriately modified Viterbi algorithm. When compared with Karect, Fiona and Coral, three current algorithms capable of correcting insertion, deletion and substitution errors, PREMIER-indel exhibits superior accuracy across a range of datasets.

4.1 Introduction

High-throughput sequencing technologies play a vital role on the frontiers of biological research, thanks to technological advancements that have driven down sequencing costs in recent years. Affordability aside, the success of sequencing-based bioinformatics, genomic and genetic analyses hinges on the acquisition of accurate, high-quality nucleotides from sequencers and upstream pipelines, with mounting studies showing that errors in high-throughput sequencing

data interfere with common downstream applications, for instance the identification of genetic variants [Kinde et al. (2011)], genome assembly [Salzberg et al. (2012)], and quantification of transcripts [Le et al. (2013)].

Despite the steady advancement in sequencing biochemistry, as well as new iterations of optical and semiconductor components that yield longer reads at higher accuracy, the quest to improve data quality has never ceased. The error rate of next-generation sequencing data is estimated to range from 0.1% to 10% [Quail et al. (2012b)], much higher than Sanger sequencing [Hoff (2009)]. Even the sequencers with the lowest error rate, like Illumina HiSeq and Ion Torrent PGM, produce datasets with a hefty volume of sequencing errors, especially after taking the high-throughput into account. It is therefore of grave importance to identify and remove sequencing errors during the primary analysis, a practice that has been demonstrated to benefit downstream analyses [Alic et al. (2016)].

To deal with noise in sequencing data, a plethora of error correction methods have been developed, targeting multiple sequencing platforms, each of which exhibits its own unique profile of sequencing bias and errors [Yang et al. (2013); Molnar and Ilie (2014); Laehnemann et al. (2015); Alic et al. (2016)]. The majority of existing tools are designed for the popular Illumina platform; many correct substitution errors only, the predominant error type produced by Illumina sequencers. In contrast, few methods can tackle insertion and deletion errors (collectively referred to as indels) [Laehnemann et al. (2015); Alic et al. (2016)], despite the multifold importance to recover those errors. Accurate detection of indel variants is imperative in clinical genomics [Hwang et al. (2015)], as many indel polymorphisms map to functionally important regions in human genomes [Mullaney et al. (2010)]. However, reliably identifying true indels is challenging [O'Rawe et al. (2013)] and is highly susceptible to the presence of indel errors introduced by the sequencer [Bragg et al. (2013)]. Secondly, several next-generation sequencing platforms, *e.g.* 454 pyrosequencing, Ion Torrent PGM and PacBio SMRT, primarily produce indel errors [Laehnemann et al. (2015); van Dijk et al. (2014)]. Indel errors are also found in Illumina sequencing data, albeit less frequent than substitutive errors. Therefore, an indel-aware error corrector certainly has broader appeal to improve indel detection and to offer an error correction tool that is less platform-dependent.

In this article, we revisit our Hidden Markov model previously developed in chapter 2 to correct only substitution errors, and propose model extensions to the state space and transition distributions to effectively handle insertion and deletion errors. This updated model, PREMIER-indel, is generally applicable to sequencing-by-synthesis (SBS) next-generation sequencing platforms, which utilize DNA polymerases to incorporate nucleotides against single-stranded DNA templates, while detecting the signals released by base synthesis. To present our work, the remaining sections of this article are arranged as follows. In section 4.2 and 4.3, we describe our algorithm, PREMIER-indel, for error correction of substitution and indel errors in high-throughput sequencing data. Specifically, in section 4.2, we model the individual reads as independent emissions from a Hidden Markov model, and discuss, in section 4.3, we discuss the procedure to estimate model parameters, modeling choices and implementation detail. In section 4.5, we study PREMIER-indel's error correction performance on multiple Illumina and Ion Torrent sequencing datasets. Our empirical results demonstrate improvements over the state-of-the-art error correction techniques, Karect [Allam et al. (2015)], Fiona [Schulz et al. (2014)] and Coral [Salmela and Schröder (2011)].

4.2 Methods

In this section, we introduce a flexible and general framework to correct sequencing errors, independent of the sequencing platform. The versatility of the model is facilitated by the fact that error correction intercepts the output of the combined sequencer/base-caller duo, which is of universal format regardless of sequencer type. For simplicity, in this section, we collectively refer to the sequencer and base-caller pair as "the sequencer." The starting point to our modeling approach is to recognize that most high-throughput sequencers (with the notable exception of nanopore sequencing) utilize DNA polymerase/ligases to incorporate nucleotides against numerous single-stranded DNA templates [van Dijk et al. (2014)], produced by random fragmentation of the genome, $\mathcal{G}$. Let $\mathbf{s}_i$ denote the i-th genomic fragment to be synthesized. Signals produced by base synthesis along $\mathbf{s}_i$ (fluorescent intensities in Illumina and PacBio, pH value in Ion Torrent) are detected and converted into nucleotides $\mathbf{x}_i$ and accompanying quality scores, $\mathbf{y}_i$, measuring the confidence of base-calls. Without loss of generality, we assume that

each base of $\mathbf{x}_i$ is called individually at discrete timepoints, or "cycles." Each discrete timepoint t produces a nucleotide on the read, denoted by $\mathbf{x}_{i[t]}$. Between each base-call, the enzyme may synthesize zero, one, or more nucleotides. Indel errors originate from the desynchronization between base synthesis and base-calling.

To mimic the above process, we employ a generative model, specifically a Hidden Markov model (HMM), which allows us to model the random process of base synthesis as a Markov chain. For the i-th fragment, the Markov process is a sequence of hidden states

$$\boldsymbol{S}_i = \boldsymbol{S}_{i,k}, \boldsymbol{S}_{i,k+1}, \ldots, \boldsymbol{S}_{i,l_i}, \tag{4.1}$$

where the subscripts $k, \ldots, l_i$ indicate the discrete timepoints. Without indel errors, all states are $k$mers, creating a k-th order Markov chain to harness the local dependence in genomic sequences. Since the initial state of the Markov chain is a $k$mer, which has initial state probability $\mu(\widetilde{\boldsymbol{S}_{i,k}})$, the process starts at time k; the first k base syntheses are lumped together into the initial state. Transitions between adjacent states (*e.g.* $\boldsymbol{S}_{i,t}$ and $\boldsymbol{S}_{i,t+1}$) correspond to the progression of DNA synthesis, with each $k$mer-to-$k$mer transition equal to the incorporation of a new base. We add special states to the state space of the HMM to handle desynchronization during synthesis (cf. §4.2.1.1). The HMM generates the observed read $(\mathbf{x}_i, \mathbf{y}_i)$ from $\boldsymbol{S}_i$ via the emission distribution, which serves dual purposes: to reduce the computational complexity of the model and to capture subtle, sequencer-specific error properties. Specifically, the initial state $\boldsymbol{S}_{i,k}$, which is assumed to be a $k$mer, emits the first k bases $\mathbf{x}_{i,k}$ and quality scores $\mathbf{y}_{i,k}$. Each subsequent state $\boldsymbol{S}_{i,t}, t > k$ emits a single base and quality score $(\mathbf{x}_{i[t]}, \mathbf{y}_{i[t]})$, regardless of the number of bases synthesized between timepoint $t-1$ and t. The latest base synthesized by the polymerase maps to the last base of the current state. The components of the HMM are discussed in detail in the following section.

4.2.1 Hidden Markov modeling of error correction

4.2.1.1 State space

The state space of the HMM consists of k^+-mers (substrings of length k or longer) and can be partitioned into three sets. First, we denote $\mathcal{K}_0$ as the set of all observed $k$mers, plus

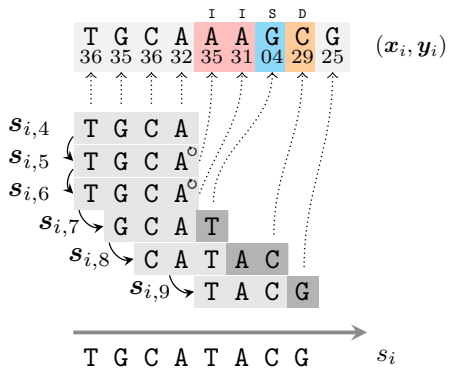


Figure 4.1: A schematic example demonstrating how PREMIER-indel models different types of error through transition and emission distributions.

their reverse complement (denoted by $\overline{\boldsymbol{\omega}}$ for $k$mer $\boldsymbol{\omega}$) in $\mathcal{R}$. Next, for every $k$mer $\boldsymbol{\omega} \in \mathcal{K}_0$, let $\boldsymbol{\omega}^{\circlearrowleft}$ denote the *insertion copy* of $\boldsymbol{\omega}$, which communicates the event of an insertion error, *i.e.* when no bases are synthesized in the cycle. Accordingly, let $\mathcal{K}_0^{\circlearrowleft} = \{\boldsymbol{\omega}^{\circlearrowleft} : \boldsymbol{\omega} \in \mathcal{K}_0\}$ denote the set of all insertion copies. Finally, to register the event of synthesizing multiple bases, we consider the sets of all $(k+m)$-mers in $\mathcal{R}$ (along with their reverse complements), $\mathcal{K}_m$, for $m = 1, 2, \ldots, m_{\max}$. If $\boldsymbol{S}_{i,t}$ is a $(k+m)$-mer, it indicates that $m+1$ bases were synthesized during the t-th cycle, while only the last base $\boldsymbol{S}_{i,t}[k+m]$ produced a base-call. Consequently, m successive bases of $\boldsymbol{S}_i$ were deleted in $\mathbf{x}_i$.

In practice, we set the state space to $\mathcal{K} \equiv \mathcal{K}_0 \cup \mathcal{K}_0{}^{\circlearrowleft} \cup \mathcal{K}_1$. We exclude $(k+m)$-mers for $m > 1$ since deletion errors are predominantly present in homopolymeric regions for most sequencing platforms [Bragg et al. (2013); Laehnemann et al. (2015)]. Successive deletion of m bases within a homopolymer can be reinterpreted as m nonconsecutive, single deletion errors so long as the original length of the homopolymer is at least $2m-1$ bases. The exclusion of longer oligomers in $\mathcal{K}$ reduces the memory footprint and model complexity without serious consequences for performance.

For completeness, define $(\boldsymbol{\omega}^{\circlearrowleft})^{\circlearrowleft} = \boldsymbol{\omega}^{\circlearrowleft}$. And for $\boldsymbol{\omega} \in \mathcal{K}_0{}^{\circlearrowleft}$, let $\boldsymbol{\omega}^{\emptyset} \in \mathcal{K}_0$ refer to the non-insertion copy of $k$mer $\boldsymbol{\omega}$.

4.2.1.2 Transition distribution

Given any $\boldsymbol{\omega} \in \mathcal{K}$, define $|\boldsymbol{\omega}|$ to be the length of $\boldsymbol{\omega}$, $\texttt{pref}_n(\boldsymbol{\omega}) \equiv \boldsymbol{\omega}_{[1\ldots n]}$ to be the n prefix of $\boldsymbol{\omega}$, and $\texttt{suff}_n(\boldsymbol{\omega}) \equiv \boldsymbol{\omega}_{[|\boldsymbol{\omega}|+1-n\ldots|\boldsymbol{\omega}|]}$ to be the n suffix. In addition, define $\mathcal{T}(\boldsymbol{\omega})$ to be the set of states to which $\boldsymbol{\omega}$ is allowed to transition, per the following rules.

a) If $\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{K} \setminus {\mathcal{K}_0}^{\circlearrowleft}$, include $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$ if and only if (i) $\texttt{suff}_{k-1}(\boldsymbol{\omega}) = \texttt{pref}_{k-1}(\boldsymbol{\nu})$ and (ii) either $\boldsymbol{\omega} \oplus \boldsymbol{\nu}_{[k\ldots|\boldsymbol{\nu}|]}$ or its reverse complement is observed at least once in $\mathcal{R}$, where $\oplus$ denotes string concatenation.

b) For $\boldsymbol{\omega} \in \mathcal{K} \setminus \mathcal{K}_1$, we let $\mathcal{T}(\boldsymbol{\omega})$ include $\boldsymbol{\omega}^{\circlearrowleft}$, so that a $k$mer can always transition to the insertion copy of itself. Successive self-transitions indicate successive insertion errors.

c) For $\boldsymbol{\omega} \in {\mathcal{K}_0}^{\circlearrowleft}$, let $\mathcal{T}(\boldsymbol{\omega}) = \mathcal{T}\left(\boldsymbol{\omega}^{\emptyset}\right) \setminus \mathcal{K}_1$, so a deletion cannot immediately follow an insertion.

The rules forbid transition from any $(k+1)$-mer to an insertion copy $k$mer, or vice versa, since such events are equivalent to a substitution error and can be handled by the emission distribution.

The transition between two states $\boldsymbol{\omega}$ and $\boldsymbol{\nu}$, provided that $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$, is governed by probability $p(\boldsymbol{\nu} \mid \boldsymbol{\omega})$. In the absence of indel errors, *i.e.* $\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{K}_0$, we consider the *genomic transition probability* $q(\boldsymbol{\nu} \mid \boldsymbol{\omega})$, with equality constraint $\sum_{\boldsymbol{\nu} \in \mathcal{T}_0(\boldsymbol{\omega})} q(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = 1$, where $\mathcal{T}_0(\boldsymbol{\omega}) = \mathcal{T}(\boldsymbol{\omega}) \cap \mathcal{K}_0$. Ideally, $q(\cdot \mid \cdot)$ is a $|\mathcal{K}_0| \times 4$ sparse matrix with non-zero entries corresponding to true transitions in the genome. To model insertion and deletion errors, the transition probability is the following mixture distribution:

$$p(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = \begin{cases} \Big[(1 - p_{d,n} - p_{i,n})\, \mathbb{1}_{\{\boldsymbol{\omega} \in \mathcal{K}_0\}} \\ \qquad + (1 - p_{d,d})\, \mathbb{1}_{\{\boldsymbol{\omega} \in \mathcal{K}_1\}} + (1 - p_{i,i})\, \mathbb{1}_{\{\boldsymbol{\omega} \in {\mathcal{K}_0}^{\circlearrowleft}\}} \Big] \cdot q(\boldsymbol{\nu} \mid \texttt{suff}_k(\boldsymbol{\omega})) & \text{if } \boldsymbol{\nu} \in \mathcal{K}_0 \\ \Big[p_{d,n} \cdot \mathbb{1}_{\{\boldsymbol{\omega} \in \mathcal{K}_0\}} + p_{d,d} \cdot \mathbb{1}_{\{\boldsymbol{\omega} \in \mathcal{K}_1\}} \Big] \\ \qquad \cdot\, q(\texttt{pref}_k(\boldsymbol{\nu}) \mid \texttt{suff}_k(\boldsymbol{\omega})) \cdot\ q\left(\texttt{suff}_k(\boldsymbol{\nu}) \mid \texttt{pref}_k(\boldsymbol{\nu})\right) \cdot \mathbb{1}_{\{\boldsymbol{\omega} \notin {\mathcal{K}_0}^{\circlearrowleft}\}} & \text{if } \boldsymbol{\nu} \in \mathcal{K}_1 \\ p_{i,n} \cdot \mathbb{1}_{\{\boldsymbol{\omega} \in \mathcal{K}_0\}} + p_{i,i} \cdot \mathbb{1}_{\{\boldsymbol{\omega} \in {\mathcal{K}_0}^{\circlearrowleft}\}} & \text{if } \boldsymbol{\nu} = \boldsymbol{\omega}^{\circlearrowleft} \end{cases} \tag{4.2}$$

where $p_{d,j}$ and $p_{i,j}$ are deletion and insertion error rate parameters, with $j \in \{n, i, d\}$ indicating the contexts of normal $k$mer-to-$k$mer transition, insertion error or deletion error. Note that transitions from/to $(k+1)$-mers do not introduce new transition parameters to the model, but reuse the genomic transition probabilities $q(\cdot \mid \cdot)$.

4.2.1.3 Emission distribution

The function of the emission distribution is twofold: to reduce model complexity and model substitution errors. To regulate the model complexity, define $D(\boldsymbol{\omega}, \boldsymbol{\nu})$ for $\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{K}$ as the edit distance between the two states. Then, for $t > k$, define the following emission distribution for the generation of $(\mathbf{x}_{i[t]}, \mathbf{y}_{i[t]})$ from state $\boldsymbol{S}_{i,t} = \mathbf{s}_{i,t}$.

$$f_t\left(\mathbf{x}_{i[t]}, \mathbf{y}_{i[t]} \mid \mathbf{s}_{i,t}, \mathbf{x}_{i,t-1}\right) = \mathbb{1}\{\mathbf{s}_{i,t} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})\} \cdot g\left(\mathbf{x}_{i[t]} \mid \mathbf{s}_{i,t}, \mathbf{x}_{i,t-1}\right) \cdot h\left(\mathbf{y}_{i[t]} \mid \mathbf{s}_{i,t}, \mathbf{x}_{i[t]}\right), \quad (4.3)$$

where the indicator function delimits a small neighborhood of hidden states, within edit distance $d_{i,t}$ to the observed $k$mer $\mathbf{x}_{i,t}$, that can emit the observation. The two pmfs $g(\cdot \mid \cdot)$ and $h(\cdot \mid \cdot)$ in (4.3) model sequencer-specific error characteristics, and we assume the following simple forms,

$$g\left(\mathbf{x}_{i[t]} \mid \mathbf{s}_{i,t}, \mathbf{x}_{i,t-1}\right) = g_0(\mathbf{x}_{i[t]} \mid \mathbf{s}_{i,t[|\mathbf{s}_{i,t}|]}), \quad (4.4)$$

$$h\left(\mathbf{y}_{i[t]} \mid \mathbf{s}_{i,t}, \mathbf{x}_{i[t]}\right) = \begin{cases} h_c(\mathbf{y}_{i,t[k]}) & \text{if } \mathbf{x}_{i,t[k]} = \mathbf{s}_{i,t[k]}, \mathbf{s}_{i,t} \in \mathcal{K}_0 \\ h_s(\mathbf{y}_{i,t[k]}) & \text{if } \mathbf{x}_{i,t[k]} \neq \mathbf{s}_{i,t[k]}, \mathbf{s}_{i,t} \in \mathcal{K}_0 \\ h_d(\mathbf{y}_{i,t[k]}) & \text{if } \mathbf{s}_{i,t} \in \mathcal{K}_1, \text{ and} \\ h_i(\mathbf{y}_{i,t[k]}) & \text{if } \mathbf{s}_{i,t} \in \mathcal{K}_0{}^{\circlearrowleft}. \end{cases} \quad (4.5)$$

Specifically, the base emission component (4.4) accounts for the bias in emitting nucleotides, which can be caused by the overlap of fluorescent spectra in Illumina sequencing [Cacho et al. (2015)], or the periodic flow cycle in the Ion Torrent sequencer [Bragg et al. (2013)]. The quality score component (4.5) models both the sequencer and the base caller and uses four distinct pmfs, conditional on error-free, substitution, deletion and insertion error scenarios.

4.2.1.4 Double-stranded dependence

In §4.2.1.2, we treat all transition probabilities $q(\cdot \mid \cdot)$ as free parameters of the model, although the inherent double-strandedness of the genomic sequence implies dependency between complementary transitions on opposite strands. In chapter 2, we introduced a labeling scheme to assign label 1 or 2 to genomic transition probabilities, so that only transition probabilities with label 1 are *free* parameters, whereas those with label 2 are dependent parameters, *i.e.*, functions of label 1 parameters. Briefly, let $\mu(\boldsymbol{\omega})$ denote the probability that any read covers $k$mer $\boldsymbol{\omega}$. Under the assumption of uniform sequencing coverage, we have $\mu(\boldsymbol{\omega}) = \mu(\overline{\boldsymbol{\omega}})$. If we let $\widetilde{\boldsymbol{\omega}} = \min\{\boldsymbol{\omega}, \overline{\boldsymbol{\omega}}\}$ (lexical order) denote the canonical $k$mer of the pair, it follows that $\mu(\widetilde{\boldsymbol{\omega}}) = \mu(\boldsymbol{\omega}) = \mu(\overline{\boldsymbol{\omega}})$. Moreover, if $k$mer transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ uniquely exists in one strand, then $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$ can be uniquely localized on the complementary strand. Then,

$$\mu(\widetilde{\boldsymbol{\omega}}) \cdot q(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = \mu(\widetilde{\boldsymbol{\nu}}) \cdot q(\overline{\boldsymbol{\omega}} \mid \overline{\boldsymbol{\nu}}). \tag{4.6}$$

With appropriate labeling, the two genomic transition probabilities in (4.6) receive opposite labels, so all label 2 transitions appear as a function of label 1 transition for all $\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{K}_0$.

$$q_2(\overline{\boldsymbol{\omega}} \mid \overline{\boldsymbol{\nu}}) = \frac{\mu(\widetilde{\boldsymbol{\omega}}) \cdot q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})}{\mu(\widetilde{\boldsymbol{\nu}})}, \tag{4.7}$$

More generally, (4.7) holds even when transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ is not unique in the genome. In chapter 2, we proposed a heuristic labeling algorithm that guarantees opposite labels to transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ and $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$, which not only reduces the model parameters by half, but also achieves good performance in practice.

4.3 Parameter estimation

Let $\boldsymbol{\theta}$ denote the vector of all model parameters. Similar to the estimating procedure used in chapter 2, we estimate the model parameters by maximizing the penalized log-likelihood function $\ell(\boldsymbol{\theta} \mid \mathcal{R}) - \rho\mathcal{J}(\boldsymbol{\theta})$ with the ℓ_0-type penalty term,

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{\boldsymbol{\omega} \in \mathcal{K}_0 : \mathcal{L}(\boldsymbol{\omega}) = 1} \sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \log\left[1 + q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})/\gamma\right]. \tag{4.8}$$

The regularization is used to induce sparsity in the transition matrix $q_1(\cdot \mid \cdot)$, which is overparameterized by erroneous transitions that can be introduced by substitution and indel errors. The effects of the penalty parameters ρ and γ are studied extensively in Supplementary Information (§A.3.4). Briefly, we prefer small γ so that the penalty function behaves like thresholding: if the expected occurrence of $k$mer transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ is less than ρ, and $|\mathcal{T}_0(\boldsymbol{\omega})| > 1$, $q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ is pushed to zero.

We employ the expectation-maximization (EM) algorithm to maximize the above penalized log-likelihood. In what follows, we discuss the expectation step and maximization step of the estimation procedure in detail.

4.3.1 E-step

The complete-data log-likelihood is,

$$\ell_c(\boldsymbol{\theta} \mid \mathcal{R}, \mathcal{S}) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{R}} \Bigg\{ \sum_{\boldsymbol{\omega} \in \mathcal{N}_{ik}(\mathbf{x}_{i,k})} \big[\log \mu(\widetilde{\boldsymbol{\omega}}) + \log f_k(\mathbf{x}_{i,k}, \mathbf{y}_{i,k} \mid \boldsymbol{\omega}) \big] + \sum_{t=k+1}^{l} \sum_{(\boldsymbol{\omega}, \boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \big[\log p(\boldsymbol{\nu} | \boldsymbol{\omega}) + \log f_t(\mathbf{x}_{i[t]}, \mathbf{y}_{i[t]} \mid \boldsymbol{\nu}, \mathbf{x}_{i,t-1}) \big] \Bigg\}, \tag{4.9}$$

where the set

$$\mathcal{N}_{it}^{\otimes}(\mathbf{x}_i) = \{(\boldsymbol{\omega}, \boldsymbol{\nu}) \in \mathcal{N}_{i,t-1}(\mathbf{x}_{i,t-1}) \times \mathcal{N}_{it}(\mathbf{x}_{i,t}) : \boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})\} \tag{4.10}$$

enumerates all allowed state transitions between cycle $t-1$ and t.

For the E-step, we need to evaluate the conditional expectation of (4.9), $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}) \equiv E[\ell_c(\boldsymbol{\theta} \mid \mathcal{R}, \mathcal{S}) \mid \mathcal{R}, \boldsymbol{\theta}^{(m)}]$, given the parameter estimates from the previous iteration m. This expected value can be partitioned into three components corresponding to the initial state

distribution, the transition distribution and the emission distribution,

$$\begin{aligned} Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) &= Q_I\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) + Q_T\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) + Q_E\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) \\ &= \sum_{i=1}^{r} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{ik}(\mathbf{x}_{i,k})} \log \mu(\widetilde{\boldsymbol{\omega}}) \cdot \zeta_{i,k}^{(m)}(\boldsymbol{\omega}) \\ &\quad + \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{(\boldsymbol{\omega}, \boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \log p(\boldsymbol{\nu} \mid \boldsymbol{\omega}) \cdot \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) \\ &\quad + \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \log f(\mathbf{x}_i[t], \mathbf{y}_i[t] \mid \boldsymbol{\omega}, \mathbf{x}_{i,t-1}) \cdot \zeta_{i,t}^{(m)}(\boldsymbol{\omega}), \end{aligned} \tag{4.11}$$

where $\xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu})$ and $\zeta_{i,t}^{(m)}(\boldsymbol{\omega})$ denote the conditional probabilities

$$\xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) = P\left(\boldsymbol{S}_{i,t-1} = \boldsymbol{\omega}, \boldsymbol{S}_{i,t} = \boldsymbol{\nu} \mid \mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}^{(m)}\right), \tag{4.12}$$

$$\zeta_{i,t}^{(m)}(\boldsymbol{\omega}) = P\left(\boldsymbol{S}_{i,t} = \boldsymbol{\omega} \mid \mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}^{(m)}\right). \tag{4.13}$$

We can compute (4.12) and (4.13) efficiently using the Forward-Backward algorithm [Rabiner (1989)].

4.3.2 M-step

Maximizing the penalized log-likelihood function can be achieved by maximizing the following objective function,

$$\widetilde{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}) = Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}) - \rho \mathcal{J}(\boldsymbol{\theta}) - \sum_{\boldsymbol{\omega} \in \mathcal{K}_0 : \mathcal{L}(\boldsymbol{\omega}) = 1} \lambda_{\boldsymbol{\omega}} \left(\sum_{\boldsymbol{\nu} \in \mathcal{T}_0(\boldsymbol{\omega})} q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) - 1 \right), \tag{4.14}$$

where the last term includes Lagrange multipliers for the label 1 $k$mers in order to enforce the equality constraint $\sum_{\boldsymbol{\nu} \in \mathcal{T}_0(\boldsymbol{\omega})} q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = 1$.

4.3.2.1 Updating $p_{i,\cdot}$ and $p_{d,\cdot}$

To update the insertion and deletion error rate parameters $p_{i,j}$ and $p_{d,j}$ for $j \in \{n, i, d\}$ in the M-step, we recognize that Q_I, Q_E, the penalty function $\mathcal{J}(\boldsymbol{\theta})$ and the Lagrange multiplier do not involve $p_{i,\cdot}$ and $p_{d,\cdot}$, hence maximizing the objective function is equivalent to maximize

$Q_T(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)})$ with respect to $p_{i,j}$ and $p_{d,j}$. It can be shown that $Q_T(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)})$ expands to

$$
\begin{aligned}
&Q_T(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}) = \\
&\sum_{i=1}^{r} \sum_{t=k}^{l_i - 1} \sum_{\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) \cdot \Bigg\{ \Big[\log(1 - p_{d,n} - p_{i,n}) \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0\} + \log(1 - p_{d,d}) \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_1\} \\
&\qquad + \log(1 - p_{i,i}) \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0{}^{\circlearrowleft}\} + \log q_{\cdot}(\boldsymbol{\nu} \mid \mathtt{suff}_k(\boldsymbol{\omega})) \Big] \cdot \mathbb{1}\{\boldsymbol{\nu} \in \mathcal{K}_0\} \\
&+ \Big[\log p_{d,n} \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0\} + \log p_{d,d} \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_1\} \\
&\qquad + \log q_{\cdot}\big(\mathtt{pref}_k(\boldsymbol{\nu}) \mid \mathtt{suff}_k(\boldsymbol{\omega})\big) + \log q_{\cdot}\big(\mathtt{suff}_k(\boldsymbol{\nu}) \mid \mathtt{pref}_k(\boldsymbol{\nu})\big) \Big] \cdot \mathbb{1}\{\boldsymbol{\nu} \in \mathcal{K}_1\} \\
&+ \Big[\log p_{i,n} \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0\} + \log p_{i,i} \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0{}^{\circlearrowleft}\} \Big] \cdot \mathbb{1}\{\boldsymbol{\nu} = \boldsymbol{\omega}^{\circlearrowleft}\} \Bigg\},
\end{aligned} \tag{4.15}
$$

where $q_{\cdot}(\cdot|\boldsymbol{\omega})$ is shortened notation for $q_{\mathcal{L}(\boldsymbol{\omega})}(\cdot|\boldsymbol{\omega})$. Next, if we derive the partial derivatives of $Q_T(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)})$ with respect to $p_{i,n}$ and $p_{d,n}$, and setting them to 0, we can show

$$
p_{i,n}^{(m+1)} = \frac{\sum_{i=1}^{r} \sum_{t=k}^{l_i - 1} \sum_{\boldsymbol{\omega} \in \mathcal{K}_0 \cap \mathcal{N}_{it}(\mathbf{x}_{i,t})} \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\omega}^{\circlearrowleft})}{\sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{(\boldsymbol{\omega}, \boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0\}}, \tag{4.16}
$$

$$
p_{d,n}^{(m+1)} = \frac{\sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{(\boldsymbol{\omega}, \boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0\} \cdot \mathbb{1}\{\boldsymbol{\nu} \in \mathcal{K}_1\}}{\sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{(\boldsymbol{\omega}, \boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0\}}. \tag{4.17}
$$

Similarly, we can derive the following updating formulas,

$$
p_{i,i}^{(m+1)} \propto \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{K}_0{}^{\circlearrowleft} \cap \mathcal{N}_{it}(\mathbf{x}_{i,t})} \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\omega}^{\circlearrowleft}). \tag{4.18}
$$

$$
p_{d,d}^{(m+1)} \propto \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{(\boldsymbol{\omega}, \boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_1\} \cdot \mathbb{1}\{\boldsymbol{\nu} \in \mathcal{K}_1\}. \tag{4.19}
$$

4.3.2.2 Updating $g_0(.,.)$ and $h_{\cdot}(\cdot)$

Updating the emission distribution parameters amounts to maximizing component

$$
\begin{aligned}
Q_E\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) &= \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \log f(\mathbf{x}_i[t], \mathbf{y}_i[t] \mid \boldsymbol{\omega}, \mathbf{x}_{i,t-1}) \cdot \zeta_{i,t}^{(m)}(\boldsymbol{\omega}) \\
&= \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \log g(\mathbf{x}_{i,t}[k] \mid \boldsymbol{\omega}_{[|\boldsymbol{\omega}|]}) \cdot \zeta_{i,t}^{(m)}(\boldsymbol{\omega}) \\
&\quad + \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \log h_{\cdot}(\mathbf{y}_{i,t[k]} \mid \boldsymbol{\omega}, \mathbf{x}_{i,t}) \cdot \zeta_{i,t}^{(m)}(\boldsymbol{\omega}).
\end{aligned} \tag{4.20}
$$

It easily follows, by taking the partial derivative of $Q_E(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)})$ with respect to the emission pmfs $g(\cdot)$ and $h_{\cdot}(\cdot)$, and setting them to zero, that

$$g_0^{(m+1)}(b_2|b_1) \propto \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega}_{[|\boldsymbol{\omega}|]} = b_1, \mathbf{x}_i[t] = b_2\} \cdot \zeta_{i,t}^{(m)}(\boldsymbol{\omega}), \quad b_1, b_2 \in \Omega. \tag{4.21}$$

$$h_c^{(m+1)}(\tau) \propto \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\mathbf{x}_i[t] = \boldsymbol{\omega}[k], \boldsymbol{\omega} \in \mathcal{K}_0, \mathbf{y}_i[t] = \tau\} \cdot \zeta_{i,t}(\boldsymbol{\omega}). \tag{4.22}$$

$$h_s^{(m+1)}(\tau) \propto \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\mathbf{x}_i[t] \neq \boldsymbol{\omega}[k], \boldsymbol{\omega} \in \mathcal{K}_0, \mathbf{y}_i[t] = \tau\} \cdot \zeta_{i,t}^{(m)}(\boldsymbol{\omega}). \tag{4.23}$$

$$h_d^{(m+1)}(\tau) \propto \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_1, \mathbf{y}_i[t] = \tau\} \cdot \zeta_{i,t}^{(m)}(\boldsymbol{\omega}). \tag{4.24}$$

$$h_i^{(m+1)}(\tau) \propto \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0{}^{\circlearrowleft}, \mathbf{y}_i[t] = \tau\} \cdot \zeta_{i,t}^{(m)}(\boldsymbol{\omega}). \tag{4.25}$$

4.3.2.3 Updating $q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$

Updating the genomic transition probabilities $q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ for $\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{K}_0$ is subject to the ℓ_0-regularization and the equality constraint. Specifically, we seek to maximize

$$\widetilde{Q}_T\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) = Q_T\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) - \rho \mathcal{J}(\boldsymbol{\theta}) - \sum_{\boldsymbol{\omega} \in \mathcal{K}_0 : \mathcal{L}(\boldsymbol{\omega}) = 1} \lambda_{\boldsymbol{\omega}} \left(\sum_{\boldsymbol{\nu} \in \mathcal{T}_0(\boldsymbol{\omega})} q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) - 1 \right). \tag{4.26}$$

Recalling that the transition probability $p(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ for $\boldsymbol{\nu} \in \mathcal{K}_1$ is a composite of two consecutive $k$mer-to-$k$mer transitions and that $q_2(\overline{\boldsymbol{\omega}} \mid \overline{\boldsymbol{\nu}}) = \frac{\mu(\widetilde{\boldsymbol{\omega}}) \cdot q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})}{\mu(\widetilde{\boldsymbol{\nu}})}$, we can show that,

$$\begin{aligned}
\tilde{Q}_T\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) = \sum_{i=1}^{r} \sum_{t=k+1}^{l_i} \sum_{(\boldsymbol{\omega},\boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \Bigg[& \log(1 - p_{d,n} - p_{i,n}) \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0\} + \log(1 - p_{d,d}) \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_1\} \\
& + \mathbb{1}\{\mathcal{L}(\mathtt{suff}_k(\boldsymbol{\omega})) = 1\} \cdot \log q_1(\boldsymbol{\nu} \mid \mathtt{suff}_k(\boldsymbol{\omega})) \\
& + \mathbb{1}\{\mathcal{L}(\mathtt{suff}_k(\boldsymbol{\omega})) = 2)\} \cdot \left(\log q_1(\overline{\mathtt{suff}_k(\boldsymbol{\omega})} \mid \overline{\boldsymbol{\nu}}) - \log \mu(\widetilde{\mathtt{suff}_k(\boldsymbol{\omega})}) + \log \mu(\widetilde{\boldsymbol{\nu}}) \right) \Bigg] \cdot \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) \cdot \mathbb{1}\{\boldsymbol{\nu} \in \mathcal{K}_0\}
\end{aligned} \tag{4.27}$$

$$\begin{aligned}
+ \sum_{i=1}^{r} \sum_{t=k}^{l_i - 1} \sum_{(\boldsymbol{\omega},\boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \Bigg[& \log p_{d,n} \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_0\} + \log p_{d,d} \cdot \mathbb{1}\{\boldsymbol{\omega} \in \mathcal{K}_1\} \\
& + \mathbb{1}\{\mathcal{L}(\mathtt{suff}_k(\boldsymbol{\omega})) = 1\} \log q_1(\boldsymbol{\nu} \mid \mathtt{suff}_k(\boldsymbol{\omega})) + \mathbb{1}\{\mathcal{L}(\mathtt{pref}_k(\boldsymbol{\nu})) = 1\} \log q_1(\mathtt{suff}_k(\boldsymbol{\nu}) \mid \mathtt{pref}_k(\boldsymbol{\nu})) \\
& + \mathbb{1}\{\mathcal{L}(\mathtt{suff}_k(\boldsymbol{\omega})) = 2)\} \left(\log q_1(\overline{\mathtt{suff}_k(\boldsymbol{\omega})} \mid \overline{\mathtt{pref}_k(\boldsymbol{\nu})}) - \log \mu(\widetilde{\mathtt{suff}_k(\boldsymbol{\omega})}) + \log \mu(\widetilde{\mathtt{pref}_k(\boldsymbol{\nu})}) \right) \\
& + \mathbb{1}\{\mathcal{L}(\mathtt{pref}_k(\boldsymbol{\nu})) = 2\} \left(\log q_2(\overline{\mathtt{pref}_k(\boldsymbol{\nu})} \mid \overline{\mathtt{suff}_k(\boldsymbol{\nu})}) - \log \mu(\widetilde{\mathtt{pref}_k(\boldsymbol{\nu})}) + \log \mu(\widetilde{\mathtt{suff}_k(\boldsymbol{\nu})}) \right) \Bigg] \\
& \cdot \xi_{i,t}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) \cdot \mathbb{1}\{\boldsymbol{\nu} \in \mathcal{K}_1\} + c
\end{aligned} \tag{4.28}$$

$$- \rho \sum_{\substack{\boldsymbol{\omega} \in \mathcal{K}_0 : \\ \mathcal{L}(\boldsymbol{\omega}) = 1}} \sum_{\boldsymbol{\nu} \in \mathcal{T}_0(\boldsymbol{\omega})} \log[1 + q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})/\gamma] - \sum_{\substack{\boldsymbol{\omega} \in \mathcal{K}_0 : \\ \mathcal{L}(\boldsymbol{\omega}) = 1}} \lambda_{\boldsymbol{\omega}} \left(\sum_{\boldsymbol{\nu} \in \mathcal{T}_0(\boldsymbol{\omega})} q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) - 1 \right), \tag{4.29}$$

where the term c involves self-transition parameters that are constant with respect to $q_1(\cdot \mid \cdot)$. The two summands (4.27) and (4.28) correspond to transitions leading into $k$mers and $(k+1)$mers. It follows that if we evaluate the partial derivative of $\tilde{Q}_T(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)})$ with respect to

parameter $q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ for $\boldsymbol{\omega}, \boldsymbol{\nu} \in \mathcal{K}_0$ and $\mathcal{L}(\boldsymbol{\omega}) = 1$, we have

$$\begin{aligned}
&\frac{\partial}{\partial q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})} \tilde{Q}_T\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(m)}\right) \\
&= \frac{1}{q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})}\Bigg[\sum_{i=1}^{r}\sum_{t=k+1}^{l_i}\sum_{(\boldsymbol{\omega}',\boldsymbol{\nu}')\in\mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \mathbb{1}\Big\{\mathtt{suff}_k(\boldsymbol{\omega}') \in \{\boldsymbol{\omega}, \overline{\boldsymbol{\nu}}\}, \boldsymbol{\nu}' \in \{\boldsymbol{\nu}, \overline{\boldsymbol{\omega}}\}\Big\} \cdot \xi_{i,t}^{(m)}(\boldsymbol{\omega}', \boldsymbol{\nu}') \qquad (4.30)\\
&\quad + \sum_{i=1}^{r}\sum_{t=k+1}^{l_i}\sum_{(\boldsymbol{\omega}',\boldsymbol{\nu}')\in\mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \mathbb{1}\Big\{\mathtt{suff}_k(\boldsymbol{\omega}') \in \{\boldsymbol{\omega}, \overline{\boldsymbol{\nu}}\}, \mathtt{pref}_k(\boldsymbol{\nu}') \in \{\boldsymbol{\nu}, \overline{\boldsymbol{\omega}}\}\Big\} \cdot \ \mathbb{1}\{\boldsymbol{\nu}' \in \mathcal{K}_1\} \cdot \xi_{i,t}^{(m)}(\boldsymbol{\omega}', \boldsymbol{\nu}') \qquad (4.31)\\
&\quad + \sum_{i=1}^{r}\sum_{t=k+1}^{l_i}\sum_{(\boldsymbol{\omega}',\boldsymbol{\nu}')\in\mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \mathbb{1}\Big\{\mathtt{pref}_k(\boldsymbol{\nu}') \in \{\boldsymbol{\omega}, \overline{\boldsymbol{\nu}}\}, \mathtt{suff}_k(\boldsymbol{\nu}') \in \{\boldsymbol{\nu}, \overline{\boldsymbol{\omega}}\}\Big\} \cdot \ \mathbb{1}\{\boldsymbol{\nu}' \in \mathcal{K}_1\} \cdot \xi_{i,t}^{(m)}(\boldsymbol{\omega}', \boldsymbol{\nu}')\Bigg] \qquad (4.32)\\
&\quad - \rho\frac{1}{\gamma + q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})} - \lambda_{\boldsymbol{\omega}} \\
&= \frac{1}{q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})}\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \rho\frac{1}{\gamma + q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})} - \lambda_{\boldsymbol{\omega}}, \qquad (4.33)
\end{aligned}$$

where the sum in brackets, we call it $\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu})$, denotes the expected number of times transition $\boldsymbol{\omega} \to \boldsymbol{\nu}$ or $\overline{\boldsymbol{\nu}} \to \overline{\boldsymbol{\omega}}$ occurs in $\mathcal{S}$. In particular, (4.30) accounts for $k/(k+1)$mer-to-$k$mer transitions, and (4.31)-(4.32) accrue the two consecutive $k$mer transitions taken by $k/(k+1)$mer-to-$(k+1)$mer transitions. From (4.33), it is straightforward to derive the updating formula for $q_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$,

$$q_1^{(m+1)}(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = \frac{\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \gamma\lambda_{\boldsymbol{\omega}} - \rho}{2\lambda_{\boldsymbol{\omega}}} \pm \frac{\sqrt{\left(\gamma\lambda_{\boldsymbol{\omega}} + \rho - \widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu})\right)^2 + 4\gamma\lambda_{\boldsymbol{\omega}}\left(\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu})\right)}}{2\lambda_{\boldsymbol{\omega}}}. \qquad (4.34)$$

The determination of the "$\pm$" sign and Lagrange multiplier $\lambda_{\boldsymbol{\omega}}$ in (4.34) is discussed at length in the Supplementary Information (§A.3.2), and is omitted here for brevity.

4.4 Error correction algorithm

To correct erroneous reads, PREMIER-bc utilizes the Viterbi algorithm to identify the state sequence $\mathbf{s}_i$ that maximizes the posterior probability $P(\mathbf{s}_i \mid \mathbf{x}_i, \mathbf{y}_i; \widehat{\boldsymbol{\theta}})$, where $\widehat{\boldsymbol{\theta}}$ refers to the estimated parameters from the EM algorithm.

We identify two weaknesses with a direct application of the Viterbi algorithm for error correction. Since PREMIER-bc assumes that the first state of the Markov chain is always a

$k$mer, indel errors within the first k bases cannot be corrected with a 5' to 3' Viterbi decoding. To supply upstream contexts that are critical for indel error removal, we adopt a two-step algorithm that runs the Viterbi algorithm in both directions, *i.e.* from 5' to 3', and then from 3' to 5', but on the complementary strand. Given the observed read $(\mathbf{x}_i, \mathbf{y}_i)$, the algorithm first identifies sequence $\widehat{\mathbf{s}}_i$ with the Viterbi algorithm. Next, we apply the Viterbi algorithm to $\overline{\widehat{\mathbf{s}}_i}$, the reverse complement of $\widehat{\mathbf{s}}_i$, without quality scores. This is because after the first round of error correction, the quality scores no longer faithfully communicate the probabilities of errors for each updated nucleotide. And if $\mathbf{x}_i$ contains indel errors, bases in $\widehat{\mathbf{s}}_i$ are likely shifted at some point, and therefore misalign with the quality scores. Accordingly, in the second pass, we use a slightly modified emission distribution with the quality score component $h_j(\cdot)$ removed. The Viterbi decoding in the reverse direction produces $\overline{\widehat{\mathbf{s}}'_i}$. Simply evaluating the reverse complement of $\overline{\widehat{\mathbf{s}}'_i}$ again, yields the final estimated sequence $\widehat{\mathbf{s}}'_i$ (i.e. the error corrected read.)

The second weak point is specifically connected to the Ion Torrent platform. We found that Ion Torrent sequencers produce reads with long stretches of successive insertion errors. For instance, in dataset $\mathcal{D}_3$, over 9% of the total insertion errors are localized in runs of successive insertions over 10 bases. Modeling these long insertions with large edit distance parameters d_{it} can be computationally intensive, especially during the EM algorithm which runs iteratively. Fortunately, only less than 1% of the reads in $\mathcal{D}_2$ contain these unusually long bursts of insertion errors, thus they are unlikely to introduce non-negligible bias to the parameter estimation. Still, failure to recover these long insertions may significantly impair the error correction performance. To address this issue, we consider another modification to the emission distribution when dealing with Ion Torrent datasets. Specifically, the indicator function in Eq. (4.3) is changed to

$$\mathbb{1}\Big\{\{\mathbf{s}_{i,t} \in \mathcal{K}_0{}^{\circlearrowleft}\} \text{ or } \{\mathbf{s}_{i,t[k]} = \mathbf{x}_{i[t]}\} \text{ or } \{\mathbf{s}_{i,t} \in \mathcal{N}_{it}(\mathbf{x}_{i,t}))\}\Big\}, \tag{4.35}$$

which becomes non-zero only if the current state $\mathbf{s}_{i,t}$ is an insertion-copy, or if no substitution error is emitted, or if $\mathbf{s}_{i,t}$ is in the edit distance neighborhood of the observed $k$mer $\mathbf{x}_{i,t}$. An example demonstrating this modified emission constraint is Fig. 4.2, which shows how the

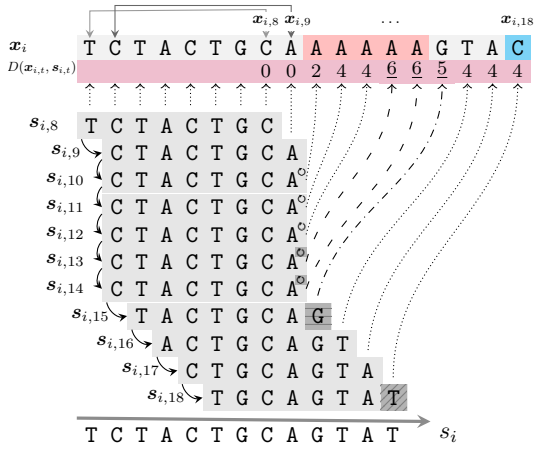


Figure 4.2: A modified emission distribution edit distance constraint for modeling long insertion errors in Ion Torrent platforms

Example with $k = 8, d_{it} = 4$ where there are insertion errors at positions 10 - 14, but no errors prior to position 10. We show a possible state sequence encountered in the decoding. States $\mathbf{s}_{i,10} - \mathbf{s}_{i,12}$ are self-transitions with $D(\mathbf{s}_{i,12}, \mathbf{x}_{i,12}) = 4$. However, $D(\mathbf{s}_{i,13}, \mathbf{x}_{i,13}) = 6$ and such a state would be disallowed under Eq. (4.3). However, it is allowed under the modified Eq. (4.35). Similarly, after decoding the insertion burst, $D(\mathbf{s}_{i,14}, \mathbf{x}_{i,14}) = 6$, the next few transitions have to be such that $\mathbf{s}_{i,t}, 15 \leq t \leq 17$ emits the observed base. A substitution error is allowed at $t = 18$ as $D(\mathbf{s}_{i,18}, \mathbf{x}_{i,18}) = 4$.

Viterbi algorithm corrects successive insertion of nucleotide A, even when the edit distance is clearly above the constraint.

4.5 Experimental Results

4.5.1 Benchmarking datasets

To compare PREMIER-indel's error correction performance with other state-of-the-art methods capable of correcting indel errors, including Karect [Allam et al. (2015)], Fiona [Schulz et al. (2014)] and Coral [Salmela and Schröder (2011)], we selected four Illumina and Ion Torrent sequencing datasets in our comparison, the details of which are listed in Table 4.1.

To determine the "ground truth" errors and quantify the error rate of each individual dataset, we aligned the reads to their respective reference genomes using the BWA aligner (v0.7.10). Specifically, the reference sequence is indexed with `bwa index`, using the default

Table 4.1: The Illumina and Ion Torrent sequencing datasets used for performance analysis

Dataset	Sequencer	Reference genome	Coverage	Error rate	Read length*	$\lvert\mathcal{R}\rvert$	Source/ Accession
$\mathcal{D}_1$	Ion Torrent	*E. coli* NC_010473.1 (4,686,137)	8×	1.48%	16-107 (92)	390,976	ERR039477*
$\mathcal{D}_2$			34×	0.94%	12-636 (324)	494,921	B22-730†
$\mathcal{D}_3$			30×	0.95%	25-629 (367)	385,452	Ion 520 Chip *E. coli* 400bp Run†
			10×	0.94%	25-588 (368)	127,687	
			5×	0.95%	25-509 (368)	63,843	
$\mathcal{D}_4$	Illumina	*S. aureus* NC_010079.1 (2,903,081)	45×	3.85%	101	1,294,104	SRR022868‡

*: for Ion Torrent datasets, the numbers are the **minimum-maximum (average)** read length.
†: available from the Ion Torrent website (http://ioncommunity.lifetechnologies.com/welcome).
‡: a subsampled version of the original SRR022868 dataset used in [Salzberg et al. (2012)] and [Allam et al. (2015)].

options; the subsequent alignment was performed using the BWA-MEM algorithm through command `bwa mem -L 100,100 <genome> <fastq> > <sam>`. The alignment parameter "`-L`" specifies the clipping penalty at both ends of the reads. We used $100, 100$ instead of the default value $5, 5$ to subdue the aggressive clipping of noisy termini of reads by BWA. Otherwise, sequencing errors in the clipped regions would not be reported by the aligner, leading to underestimated error rate, and underdetermined ground truth errors. We assume that all mismatches between the reference genome and the reads are bona fide sequencing errors.

For some datasets, we carried out preprocessing of the original data prior to error correction. Dataset $\mathcal{D}_3$ originally had high sequencing coverage (584×), for which we created three randomly subsampled datasets at coverage levels 30×, 10× and 5×. The decreasing coverage of the three subsamples allows us to measure the sensitivity to sequencing coverage for each method in comparison. The Illumina dataset $\mathcal{D}_4$ contains ambiguous base-calls ("N" bases) which are currently not modeled by PREMIER-indel. Therefore, before running PREMIER-indel on dataset $\mathcal{D}_4$, for any given read $\mathbf{x}_i$ in $\mathcal{R}$, we convert each N base in $\mathbf{x}_i$ to a random nucleotide in $\{A, C, G, T\}$, if the number of N bases is less than 25% of the read length. Otherwise, the N bases will be left intact, and PREMIER-indel will skip $\mathbf{x}_i$ during the EM algorithm, and subsequent error correction. We note that these skipped errors remain in the dataset and are still counted against PREMIER-indel during the benchmarks.

4.5.2 Performance metrics

To benchmark the performance of individual method, we can view error correction as a binary classification problem at the base level. Specifically, an instance of sequencing error, if fully recovered by error correction, is registered as a true positive (TP). Similarly, define false negative (FN) for an uncorrected error, false positive (FP) for a new error introduced by the error corrector, and true negative (TN) for an untouched correct base.

For any given read sequence $\mathbf{x}_i$, the per-read statistics $(\mathrm{TP}_i, \mathrm{FP}_i, \mathrm{FN}_i, \mathrm{TN}_i)$ are tallied as follows. Suppose that $\mathbf{x}_i$ aligns to position τ_i on the genome with minimum edit distance $d_{\tau_i}(\mathbf{x}_i)$, and that the error corrected sequence is $\widehat{\mathbf{x}}_i$. We then compute the alignment of $\widehat{\mathbf{x}}_i$ to genome position τ_i, and let $d_{\tau_i}(\widehat{\mathbf{x}}_i)$ denote the edit distance of the alignment. [Allam et al. (2015)] showed that, $d_{\tau_i}(\mathbf{x}_i) = \mathrm{TP}_i + \mathrm{FN}_i$, $d_{\tau_i}(\widehat{\mathbf{x}}_i) = \mathrm{FP}_i + \mathrm{FN}_i$. Using the two edit distances, we can compute the per-read gain metric [Yang et al. (2013)]

$$\mathbf{gain}_i = (\mathrm{TP} - \mathrm{FP})/(\mathrm{TP} + \mathrm{FN}) = \frac{d_{\tau_i}(\mathbf{x}_i) - d_{\tau_i}(\widehat{\mathbf{x}}_i)}{d_{\tau_i}(\mathbf{x}_i)} \in (-1, 1), \tag{4.36}$$

which reflects the effective percentage of errors corrected in read $\mathbf{x}_i$. Aggregating the statistics over all *mappable* reads, we can compute the gain for the entire dataset. In addition, for each corrected dataset, we also compute the post-error-correction alignment rate to quantify the extent to which sequence alignment can be improved by error correction.

4.5.3 Software comparison

To generate error corrected reads from Coral, Fiona and Karect, we ran these software with their default parameters, except for the following adjustments. For Fiona, we provided the error rate (`-e`) and genome length (`-g`) parameters as the values listed in Table 4.1. On account of the higher indel error rate in Ion Torrent data, we also set Fiona's maximum indel length parameter (`-id 4`) to 4, the largest allowable value, for all Ion Torrent datasets. To run Karect, we set `-matchtype=edit` so that substituion, insertion and deletion errors have equal costs in the alignment, and `-celltype=haploid`, since all reference organisms have haploid genomes. We ran Coral in its "`-454`" mode, which sets appropriate alignment scores to allow all three types of errors to be corrected in both Illumina and Ion Torrent datasets.

For PREMIER-indel, we use $k = 20, \gamma = 10^{-20}$ for all datasets. We set $d_{ik} = 4$ for allowing up to 4 substitution errors within the first observed $k$mer, and set $d_{it} = 10$ for $t > k$. Note that in Ion Torrent datasets, successive insertions of d_{it} nucleotides or longer are allowed. The penalty threshold parameter ρ was determined based on coverage (see Supplementary Information, §A.5.1), and we set $\rho = 1$ for $\mathcal{D}_1$ and $\mathcal{D}_{3,5\times}$, $\rho = 2$ for $\mathcal{D}_{3,10\times}$, and $\rho = 4$ for $\mathcal{D}_2, \mathcal{D}_{3,30\times}$ and $\mathcal{D}_4$.

Table 4.2 summarizes the error correction performance of the four methods in comparison, across the four datasets. Both Karect and PREMIER-indel significantly outperform Fiona and Coral on all datasets, in terms of data quality at the base level (gain) and at the read level (proportion of error-free reads.) Between the two front-runners, PREMIER-indel outmatches Karect on Ion Torrent dataset $\mathcal{D}_1$ with sizeable margin on base qualities, and produces more reads without errors. On the Illumina dataset $\mathcal{D}_4$, PREMIER-indel leads in the gain metric, but has 1.36% less error-free reads. Karect performs better on $\mathcal{D}_2$ and $\mathcal{D}_3$ when coverage is adequate, but its error correction performance is more sensitive to the drop in sequencing coverage than PREMIER-indel, as evident by the faster degradation of base and read qualities when sequencing coverage declines from 30× to 5×: PREMIER-indel regains the lead on read-wise quality at 10× coverage of $\mathcal{D}_3$, and at 5× coverage, PREMIER-indel ranks atop on all metrics. We attribute PREMIER-indel's better performance at low coverage level to its probabilistic nature, allowing PREMIER-indel to model more subtle information in the sequencing data.

Our model is known to have limited error correction capability within the first k bases due to the lack of upstream contexts, and the smaller distance constraint. Our assumption that the first k nucleotides are free of indel errors can also negatively impact PREMIER-indel's performance. To measure the extent to which PREMIER-indel underperforms due to sequencing errors within the first observed $k$mer, Table 4.3 compares the error correction performance of Karect and PREMIER-indel on reads whose first k nucleotides are free of substitution, insertion and deletion errors. The observation that PREMIER-indel performs better on almost all metrics over all datasets reveals the relative strength of our probabilistic methods, especially when context information is provided. Meanwhile, it also highlights the weakness of our

Table 4.2: Performance comparison of PREMIER-indel (PMRind), Coral, Karect and Fiona on benchmarking datasets

Dataset		Method	Alignment rate	Gain	Error-free Reads% Before	After
$\mathcal{D}_1$		**PMRind**	99.50%	**0.7317**	54.00%	**87.80%**
		Karect	99.58%	0.6259		85.27%
		Fiona	**99.63%**	0.4063		79.54%
		Coral	99.52%	0.3727		73.24%
$\mathcal{D}_2$		**PMRind**	98.15%	0.9086	20.32%	86.77%
		Karect	98.21%	**0.9308**		**89.46%**
		Fiona	**98.35%**	0.7217		75.31%
		Coral	98.07%	0.7228		65.29%
$\mathcal{D}_3$	30×	**PMRind**	99.29%	0.9360	20.17%	92.37%
		Karect	99.28%	**0.9686**		**94.98%**
		Fiona	**99.30%**	0.7637		78.29%
		Coral	99.28%	0.8347		70.17%
	10×	**PMRind**	99.26%	0.9239	20.12%	**90.69%**
		Karect	99.26%	**0.9313**		88.64%
		Fiona	99.26%	0.7024		73.47%
		Coral	99.25%	0.8574		74.63%
	5×	**PMRind**	99.31%	**0.8329**	20.03%	**78.59%**
		Karect	99.30%	0.8311		75.87%
		Fiona	99.31%	0.5639		58.41%
		Coral	99.30%	0.7317		62.64%
$\mathcal{D}_4$		**PMRind**	90.08%	**0.8039**	30.52%	80.91%
		Karect	**92.04%**	0.7716		**82.27%**
		Fiona	91.07%	0.4738		69.49%
		Coral	89.55%	0.1707		51.00%

model, as it is highly sensitive to presence of errors within the first k positions, and calls for improvement in the future for better construction of the first neighborhoods.

4.6 Discussion

In this work, we developed PREMIER-indel, a probabilistic error correction method that corrects substitution, insertion and deletion errors. The main idea of PREMIER-indel is to, with some degree of generalization, model the operation of SBS sequencers as a Hidden Markov model. PREMIER-indel builds on top of our previous substitution-only error corrector, PREMIER, by extending the state space of the HMM with special states to model the desynchronization between base synthesis and base-calling. More specifically, we introduced an insertion-copy of $k$mers to model insertion errors, and $(k+m)$-mers to handle deletion errors, in addition to the $k$mer-only state space in PREMIER. Using empirical analysis, we show that

Table 4.3: Performance comparison of PREMIER-indel (PMRind), and Karect, on reads with error-free first $k$mer($k = 20$)

Dataset		Method	Alignment rate	Gain	Error-free Reads% Before	After
$\mathcal{D}_1$		**PMRind**	99.48%	**0.7819**	56.85%	**90.30%**
		Karect	**99.56%**	0.6277		86.71%
$\mathcal{D}_2$		**PMRind**	**98.06%**	**0.9414**	21.39%	88.28%
		Karect	98.11%	0.9319		**89.56%**
$\mathcal{D}_3$	30×	**PMRind**	99.24%	**0.9717**	21.50%	94.02%
		Karect	99.24%	0.9689		**95.04%**
	10×	**PMRind**	99.21%	**0.9589**	21.47%	**92.33%**
		Karect	99.21%	0.9303		88.79%
	5×	**PMRind**	99.25%	**0.8888**	21.40%	**80.80%**
		Karect	99.25%	0.8277		76.36%
$\mathcal{D}_4$		**PMRind**	87.89%	**0.8820**	37.25%	**84.19%**
		Karect	**90.28%**	0.8293		82.79%

PREMIER-indel significantly improves the error correction performance over Fiona and Coral, and is competitive with Karect, the state-of-the-art error correction software using multiples-sequence alignment. We note that PREMIER-indel's performance is more robust in datasets with shorter read length and lower coverage.

The expansion of the state space, particularly the insertion-copy $k$mers, $\mathcal{K}_0{}^{\circlearrowleft}$, leads to a more computationally intensive model. Since the HMM uses $k$mers in $\mathcal{K}_0$ to model the progression of DNA templates, which are assumed to include only true genomic $k$mers (ignoring errors introduced in upstream pipelines), the transitions $\boldsymbol{\omega} \to \boldsymbol{\omega}^{\circlearrowleft}$ that model the void of base synthesis should only be allowed for genomic $k$mers as well. Recall that in §4.2.1.2, we let $\boldsymbol{\omega}^{\circlearrowleft} \in \mathcal{T}(\boldsymbol{\omega})$ for any $\boldsymbol{\omega} \in \mathcal{K} \setminus \mathcal{K}_1$. This modeling choices simplifies implementation, however incurs undesirable and excess computational overhead, by allowing every $k$mer, including erroneous ones to transition to its insertion-copy. This bottleneck can be mitigated by selectively including $\boldsymbol{\omega}^{\circlearrowleft}$ in $\mathcal{T}(\boldsymbol{\omega})$ for prescreened $\boldsymbol{\omega}$.

Our model formulation currently assumes simple forms for the desynchronization events. For instance, in the transition distribution, parameters $p_{d,j}$ and $p_{i,j}$ have no dependence on upstream contexts, although contextual information, such as homopolymeric motifs, can be highly indicative of indel errors. Similarly, the base emission probability mass function (4.4) does not condition on whether the hidden state $\boldsymbol{S}_{i,t}$ contains an indel error or not. We expect

the error correction performance of PREMIER-indel to further improve with appropriate model extensions that take account of the above well-studied error properties in the future.

CHAPTER 5. CONCLUSION

In this dissertation, we developed three probabilistic models to improve the data quality of next-generation sequencing data, by explicitly modeling the generative process of observed sequencing data (nucleotides or fluorescent intensities) from a latent base-synthesizing process as a Hidden Markov model. On top of the basic Hidden Markov model we proposed in chapter 2, which deals with substitution errors only in Illumina sequencing data, we have formulated variants of the same model through appropriate model extensions, and successfully tackled the problems of base calling for the Illumina platform and indel error correction on multiple platforms.

The main strength of statistical modeling of sequencing noise is that probabilistic models can better emulate the true generative process of high-throughput sequencers. The base-calling and error correction projects from previous chapters not only demonstrate the superior performances of a model-based strategy, but also showcase the flexibility of probabilistic methods than algorithmic counterparts. In addition, our models are capable of producing probabilistic assessments of individual base qualities in addition to the most likely outcomes. And such nuances can often be crucial in downstream analysis like variant calling, genotyping, and transcript quantification etc.

Consider our base-calling software PREMIER-bc in chapter 3 as a counterexample to the common practice of aggressive filtration and compression of data by bioinformatics tools. Prematurely discarding valuable information from expensive sequencing experiments is an irreversible act with direct impact on the downstream analysis. Ideally, our probabilistic framework can be more than an error corrector, or a base-caller, but can be extended to include existing models of secondary analyses directly. With the ever decreasing cost of computational

hardware and data storage, it is possible to envision a more streamlined bioinformatics pipeline with less transitions between stages, that maximizes the utilization of raw data.

More realistically, we identify the following areas of our probabilistic framework that may be improved in the future:

(i) In chapter 3, we revamped the emission distribution of our Hidden Markov model, using the direct emission of fluorescent intensities to model common Illumina sequencing errors. We demonstrated significant performance improvement over existing base-callers, even without explicitly modeling phasing and prephasing effects in the Illumina sequencing process, which leads to insertion and deletion errors. Naturally, the PREMIER-indel model in chapter 4 can be grafted with the emission distribution in chapter 3, to become an indel-aware base-caller for Illumina sequencing. With proper modifications, it also has the potential to become an Ion Torrent base-caller. Essentially, during each flow cycle of Ion Torrent sequencing, only one species of dNTP is added for synthesis. If the current cycle adds dATP, for example, then the current state should only be allowed to transition into $(k+m)$-mers for $m \geq 0$, where the added $m+1$ bases are exclusively homopolymers of nucleotide A. A similar idea of using state machines to base-call Ion Torrent data appeared in Golan and Medvedev (2013).

(ii) A recurring issue identified in the three projects is the lack of error correction performance within the first $k$mer of the observed sequence. More importantly, ill-constructed first neighborhoods cripple the identification of errors in subsequent positions, due to the contaminated upstream contexts, as our models always move from the 5' to 3' direction. Efficient methods for indexing $k$mers [Kowalski et al. (2015)] provide an alternative approach to construct the first neighborhoods than our current heuristics. More precisely, for a target sequence, we consider finding overlapping reads using similar tactics found in MSA-based error correction software, but only for the purposes to identify candidate $k$mers that align with the observed first $k$mer.

(iii) To enable more efficient model estimation on complex genomes, the hard limit on $k \leq 30$ should be extended, especially considering sequencers nowadays produce longer reads. For instance, in chapter 2, PREMIER's performance on the human chromosome 14 dataset was capped by $k$mer size. The proposed improvement on first neighborhood construction above is

necessary to enable using longer k in our model, since the current heuristic neighborhood construction method may not scale with large values of k.

(iv) The ℓ_0-penalty we used in chapter 2 assumes uniform coverage and unique $k$mers. However, both assumptions can be violated in practice. The same $k$mer may derive from multiple loci of in a genome, or different transcripts in a RNA-seq experiment, or even different genomes in a metagenomics study. The latter two cases may exhibit highly variable sequencing coverage at transcript-level, or genome-level as well. To relax the $k$mer uniqueness assumption, we can add multiple copies of the same $k$mer to the state space. Theoretically, each copy then corresponds to a unique context from which this copy is derived, and leads to a unique transition downstream. And since the equality constraint protects the ℓ_0-penalty from pushing the unique transition probability to zero, even low-coverage variants would survive. Determining the number of copies of each $k$mer is a difficult model selection problem. Nevertheless, we have previously implemented an algorithm that utilizes backtracking on the de Bruijn graph to estimate the number of copies for any given $k$mer. Technically, each new copy of $k$mer can be treated as special case of the insertion-copy $k$mer in chapter 4, and thus can be implemented with the same strategy.

APPENDIX A. SUPPLEMENTARY INFORMATION

A.1 k, d-local error correction method

In this section, we provide the definition for k, d-local error correction methods. Specifically, an error correction method is considered k, d-local if it:

(i) extracts the k-spectrum $\mathcal{K}$ (or more generally, the k-spectra, for a series of k^*-spectrum such that k^* is no larger than k), either explicitly or implicitly from $\mathcal{R}$, and

(ii) assumes that $\mathcal{K}$ contains *all* true $k$mers in $\mathcal{G}$, and therefore by correcting errors, it does not increase the size of $\mathcal{K}$; and

(iii) makes no more than d error corrections in any window of size k, for any given read $(\mathbf{x}_i, \mathbf{y}_i)$.

At its basic form, a qualifying k, d-local method corrects an erroneous position by considering $k$mers, $k$mer multiplicity, and quality scores within a $2k-1$ window, which centers at the error locus. More advanced methods may elect to scour the vicinity around the k $k$mers (within the $2k-1$ window) on the de Bruijn graph for additional context information. Nevertheless, the context supplemented by the graph is often limited and more local, when compared with the read-local methods, *e.g.* Karect. This locality effect is especially prominent for $k$mers derived from repetitive regions in $\mathcal{G}$, as demonstrated by the following examples in Fig. A.2.

A.1.1 Theoretically optimal performance of k, d-local methods

We seek to analyze the upper bound of error correction performance, for the class of k, d-local methods, by implementing an "oracle" that conforms to the definitions above. For a given sequencing dataset $\mathcal{R}$, the oracle is empowered by the knowledge of ground truth errors

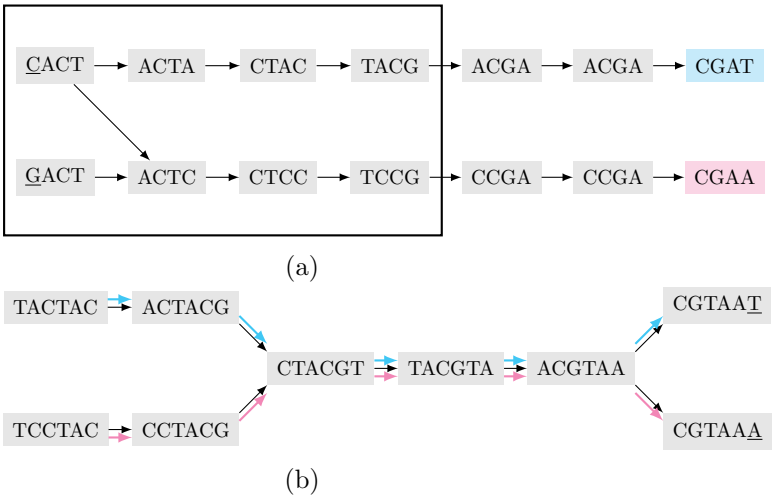


Figure A.2: Examples demonstrating how k, d-local methods may utilize (and are limited by) the de Bruijn graphs.

(a) If the kmer $CACT$ is misread as $GACT$ in a read, the error can be corrected utilizing context information on the de Bruijn graph beyond the $k-1$-local window (shown in box), as long as the paths on the graph are distinct. Suppose that the observed read is $\underline{G}ACTACGAA\underline{T}$, and that base T is of high quality score, then the optimal error-correction is to change the first G to C. (b) When the paths are not distinct, the local upstream contexts become ambiguous. Suppose that the blue and red paths corresponds to two distinct regions in the genome. Since the "color" information is not retained by the de Bruijn graph, the k, d-local methods will fail to correct the error, if the last base T is misread as an A, whereas the read-local methods, like Karect and Coral, can use alignments to potentially recover the error.

in $\mathcal{R}$. After removing all such errors, the *true* de Bruijn graph can be obtained from the error-corrected reads, $\mathcal{R}^*$. Each edge in the de Bruijn graph, representing transition between $k$mers, is weighted by the number of observed incidence in $\mathcal{R}^*$.

To correct errors in read $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{R}$, the oracle computes the best alignment of $(\mathbf{x}_i, \mathbf{y}_i)$ to the true de Bruijn graph, within the constraint that no more than d mismatches are allowed in any aligned $k$mer. The alignment is performed by first decomposing $(\mathbf{x}_i, \mathbf{y}_i)$ into overlapping $k$mers, $\mathbf{x}_{i,k}, \mathbf{x}_{i,k+1}, \ldots, \mathbf{x}_{i,l_i}$. Provided that the 5' ends of Illumina reads typically have higher quality, we assume that the oracle can correct *all* errors in $\mathbf{x}_{i,k}$, producing an error-free $k$mer $\mathbf{x}^*_{i,k}$, which is subsequently aligned to its exact copy on the de Bruijn graph. For $t \geq k$, two adjacent $k$mers $\mathbf{x}_{i,t}, \mathbf{x}_{i,t+1}$ can then be aligned to any edge on the de Bruijn graph, connecting vertices $\boldsymbol{\omega}$ and $\boldsymbol{\nu}$ ($\boldsymbol{\omega}_{[2\ldots k]} = \boldsymbol{\nu}_{[1\ldots k-1]}$), as long as $D(\mathbf{x}_{i,t}, \boldsymbol{\omega}) \leq d$ and $D(\mathbf{x}_{i,t+1}, \boldsymbol{\nu}) \leq d$ (the only exception being when $t = k, \boldsymbol{\omega} = \mathbf{x}^*_{i,k}$). Suppose that $\mathbf{x}_{i,k}, \ldots, \mathbf{x}_{i,l_i}$ aligns to a certain path on the de Bruijn graph: $\mathbf{s} = \boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_{l_i-k+1}$, the alignment is scored by,

$$\prod_{j=k+1}^{l_i} \left(10^{-\frac{\mathbf{y}_i[j]}{10}}\right)^{\mathbf{x}_i[j] \neq \mathbf{s}[j]} \cdot \left(1 - 10^{-\frac{\mathbf{y}_i[j]}{10}}\right)^{\mathbf{x}_i[j] = \mathbf{s}[j]} \times \prod_{j=1}^{l_i-k+1} w(\boldsymbol{\omega}_j, \boldsymbol{\omega}_{j+1}),$$

where $w(\boldsymbol{\omega}_j, \boldsymbol{\omega}_{j+1})$ is the weight of edge $\boldsymbol{\omega}_j \to \boldsymbol{\omega}_{j+1}$ on the graph. The alignment that scores atop is considered as the error-corrected read by the oracle.

Benefiting from the information of the true de Bruijn graph, as well as the quality scores, which the oracle utilizes to the same extent as other k, d-local methods, the oracle's performance is expected to dominate its peers, as demonstrated in Table A.1. Specifically, since the oracle utilizes the ground truth to eliminate all non-genomic erroneous $k$mers from $\mathcal{K}$ (and thus the graph), whereas k, d-local methods commonly rely on a cut-off on $k$mer multiplicity for decision-making, the oracle is extremely powerful in removing sequence-specific errors which produce erroneous $k$mers of high frequency, and is less susceptible to datasets with low-coverage or strong coverage variation, in which some true $k$mers represent with underwhelming redundancy and are classified as errors by k, d-local methods. In that regard, the oracle's performance can be overly optimistic.

Table A.1: Error-correction performance of the oracle of k, d-local methods.

		Oracle		
Dataset		**sn.**	**gain**	e_p
D1	54×	0.974	0.971	0.014
	10×	0.948	0.942	0.027
D2	21×	0.936	0.930	0.041
	5×	0.878	0.862	0.081
D3	30×	0.973	0.967	0.029
	5×	0.866	0.834	0.147

A.2 Implementation

A.2.1 Labeling transitions

To capitalize on the information available in the reverse complement strand when strandedness cannot be known, we proposed in §2.2 to assign labels to $k$mers. The labels identify *dependent* (label *2*) parameters that can be written as functions of *free* (label *1*) parameters. Specifically, for each $\boldsymbol{\omega} \in \mathcal{K}$, we require $\mathcal{L}(\boldsymbol{\omega}) \neq \mathcal{L}(\overline{\boldsymbol{\nu}})$ for all $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$. By the transitive property, we also require $\mathcal{L}(\boldsymbol{\omega}) = \mathcal{L}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu}$ with $\overline{\boldsymbol{\mu}} \in \mathcal{T}(\overline{\boldsymbol{\nu}})$ and $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$. We show that such labeling of $k$mers is equivalent to identifying and assigning distinct labels to the two parts of disjoint bipartite graphs formed from state space $\mathcal{K}$ such that nodes are $k$mers and edges are transitions.

We can partition $\mathcal{K}$ into disjoint sets of $k$mers sharing either of a pair of $(k-1)$-mers $(\boldsymbol{\delta}, \overline{\boldsymbol{\delta}})$ as suffixes. When k is even, $\boldsymbol{\delta} \neq \overline{\boldsymbol{\delta}}$, so if we define $\underline{\boldsymbol{\delta}}$ as the lexically smaller of ($\boldsymbol{\delta}$ and $\overline{\boldsymbol{\delta}}$), then each set of $k$mers is uniquely identified by $\underline{\boldsymbol{\delta}}$. Given $\underline{\boldsymbol{\delta}}$, we construct a corresponding bipartite graph $G(\underline{\boldsymbol{\delta}})$ consisting of two disjoint sets of $k$mers (or vertices):

$$U(\underline{\boldsymbol{\delta}}) = \{\boldsymbol{u} = b_1 \oplus \underline{\boldsymbol{\delta}} : \boldsymbol{u} \in \mathcal{K}\} \text{ and } V(\underline{\boldsymbol{\delta}}) = \{\overline{\boldsymbol{v}} : \boldsymbol{v} = \underline{\boldsymbol{\delta}} \oplus b_2, \boldsymbol{v} \in \mathcal{K}\},$$

where $\oplus$ is the string concatenation operator and $b_1, b_2 \in \Omega$. Edges, denoted by $E(\underline{\boldsymbol{\delta}})$, are added to $G(\underline{\boldsymbol{\delta}})$ using the observed transitions between $k$mers. Specifically, we add an undirected edge between $\boldsymbol{u} \in U(\underline{\boldsymbol{\delta}})$ and $\boldsymbol{v} \in V(\underline{\boldsymbol{\delta}})$ for every $\boldsymbol{v} \in \mathcal{T}(\boldsymbol{u})$. By construction of $\mathcal{K}$ and $\mathcal{T}(\cdot)$, the addition of edges is finished after one pass through $U(\underline{\boldsymbol{\delta}})$.

Labeling $k$mers amounts to assigning *1* or *2* to $k$mers in U and the other to V, but because the regularization achieved during penalized optimization applies directly to transitions out of $k$mers labeled *1*, but only indirectly to transitions form $k$mers labeled *2*, the labeling choice matters. In particular, the penalty cannot remove *unique* transitions even if they are errors because constraint Eq. (3.1) forces the unique transition probability identically equal to one. We therefore prefer to assign label *1* to $k$mers with more exit transitions and let the penalty identify the error transitions among the true transitions. Let the number of $k$mers with *non-unique* transitions in U and V be n_U and n_V, respectively. Then, we award label *1* to the $k$mers of U if $n_U > n_V$ or V if $n_U < n_V$. If $n_U = n_V$, we assign *1* to the smaller set of $k$mers, U if $|U| < |V|$ or V if $|V| < |U|$. When $|U| = |V|$, we award label *1* to the set with lower summed $k$mer coverage, presuming that coverage is low because some of these transitions are errors. If there is a tie in coverage, label *1* is assigned randomly to U or V. This ad hoc solution is one of many possible labeling strategies, but it seems to work well in practice.

The labeling strategy can be further refined to independently assign labels to disjoint subgraphs of each bipartite graph. If $G(\underline{\boldsymbol{\delta}})$ can be partitioned, it suggests multiple locations of $\underline{\boldsymbol{\delta}}$ in the genome, and it is desirable to label the transitions of each location separately. For example, if an error is introduced downstream of $\boldsymbol{\delta}$ at the first location, and a distinct error occurs downstream of $\overline{\boldsymbol{\delta}}$ at the other location, then labeling them together prevents direct penalization of one of the error transitions. Each bipartite graph $G(\underline{\boldsymbol{\delta}})$ can be partitioned into one or more connected components. We then apply the labeling algorithm independently to each component, $G_j(\underline{\boldsymbol{\delta}})$, of $G(\underline{\boldsymbol{\delta}})$.

To illustrate the complete labeling procedure, consider the schematic example given in Fig. A.4, featuring the labeling of $k$mers with suffixes (ACT, AGT) for $\underline{\boldsymbol{\delta}} = ACT$. We construct the bipartite graph $G(\underline{\boldsymbol{\delta}})$ from the reads in Fig. A.3(a). Each read identifies a transition with suffix/prefix match ACT or AGT. We can represent each transition as a pair of $k$mers and an edge linking them. Rearranging the collection of $k$mers and edges produces $G(\underline{\boldsymbol{\delta}})$ (Fig. A.3(b)). This particular bipartite graph can be partitioned into the red and blue bipartite graphs of Fig. A.3(c), whose $k$mers are labeled independently, using the heuristic rules described above. The red component assigns label *1* to the $k$mer in $U_1(\underline{\boldsymbol{\delta}})$ because $k$mer $AACT$ has multiple exit

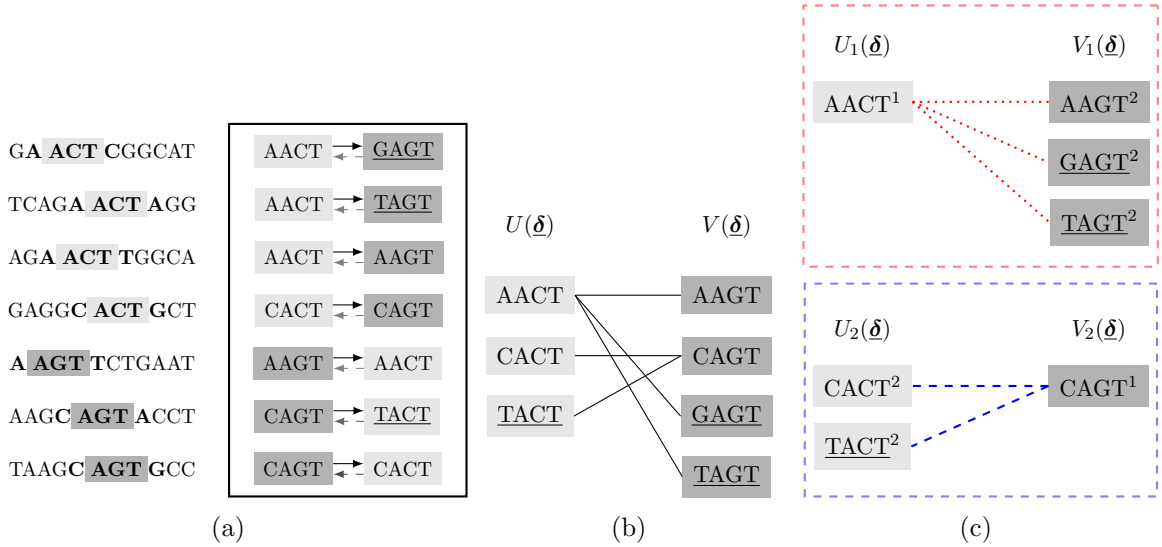


Figure A.4: A demonstration of the bipartite graph formation and labeling strategy.

(a) Extracting $k$mers and transitions (right panel) from the reads (left panel) containing $k$mers with suffixes ACT or AGT. The underlined $k$mers are those implied to exist on the reverse complement strand, though they are not observed in the reads. (b) We convert the $k$mers into vertices and transitions into edges, but group the vertices by shading to form a bipartite graph with parts, $U(\underline{\boldsymbol{\delta}})$ and $V(\underline{\boldsymbol{\delta}})$. (c) The bipartite graph in (b) factors into two disjoint components, and the labeling algorithm applied to each component separately, assigns label "1" or "2" to the $k$mers in each part.

transitions, whereas $k$mers in $V_1(\underline{\boldsymbol{\delta}})$ all have unique transitions. For the same reason, in the blue component, $V_2(\underline{\boldsymbol{\delta}})$ is assigned label *1*. Looking at the reads, it is evident that either *ACT* occurs in two genomic locations or, much less likely, some of the reads contain many coincident errors. Indeed, it is reasonable to assume that all underlined $k$mers contain errors. In this scenario, separating $G(\underline{\boldsymbol{\delta}})$ into two components allows PREMIER to subject all likely error transitions to the direct cost of the penalty, so that pathways with errors can be eliminated.

A.2.2 Construction of $k$mer neighborhoods

When implementing the Baum-Welch and Viterbi algorithms, we only consider hidden states $\mathbf{s}_{i,t} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})$ as plausible true states for $\mathbf{x}_{i,t}$. Along with constraints on the k-spectrum (2.1), allowable transitions (3.1), and our initialization scheme (§A.3.3), these neighborhoods define the structure of the HMM before it is made even sparser by penalized parameter estimation. This section describes neighborhood construction.

For tractable computation, the size of the neighborhood is position- and read-specific, and depends on the maximum Hamming distance d_{it}. By default, we let $d_{it} = \frac{k}{2}$, to ensure that PREMIER is capable of recovering exceptionally noisy reads. For reads of highly-repetitive or low-complexity regions in $\mathcal{G}$, the neighborhood size escalates rapidly with d_{it}. Even with infinite compute resources, any k, d-local method has limited ability to resolve errors in such regions, so we adjust d_{it} downward both to reduce computational complexity and false positives. Specifically, we select the maximum d_{it} such that $|\mathcal{N}_{it}(\mathbf{x}_{i,t})| \leq m_{it}$, where

$$m_{it} = \begin{cases} m_2 & \text{if all elements of } \mathbf{y}_{i,t} \leq 2 \\ m_1 & \text{otherwise.} \end{cases}$$

In practice, we let $m_1 = 2^8$, and $m_2 = 2^{15}$. This choice is specific to current Illumina technology, where quality score 2 communicates noisy read ends, but could be easily adjusted.

To further reduce the overhead incurred by constructing the first neighborhood $\mathcal{N}_{ik}(\mathbf{x}_{i,k})$ and its downstream effects, we adopted the following heuristic rule, which relies on the assumption that most reads have error-free first $k$mers. If the number of $\mathbf{x}_{i,k}$ $k$mers, $n(\mathbf{x}_{i,k})$, observed in the read set exceeds the penalty threshold ρ, we let $\mathcal{N}_{ik}(\mathbf{x}_{i,k}) = \{\mathbf{x}_{i,k}\}$. Otherwise, we

employed two methods to explicitly expand $\mathcal{N}_{ik}(\mathbf{x}_{i,k})$. In the first and preferred approach, $\mathcal{N}_{ik}(\mathbf{x}_{i,k})$ is constructed by backtracking using the initialized transitions (§A.3.3) from the first observed $k$mer $\mathbf{x}_{i,t}$ with $n(\mathbf{x}_{i,t}) > \rho$. During backtracking, all pathways are retained unless they deviate over Hamming distance $\frac{k}{2}$ from the observed $k$mers in the read $\mathbf{x}_i$. The first neighborhood $\mathcal{N}_{ik}(\mathbf{x}_{i,k})$ is the collection of all surviving $k$mers after $t-k$ backtrack steps. Under the assumption that $\mathbf{x}_{i,t}$ is *error-free* and all true transitions are observed at least once on either strand, the true $k$mer should be in the constructed set. If the read contains no $k$mers with $n(\mathbf{x}_{i,t}) > \rho$, then PREMIER utilizes a fallback approach to construct $\mathcal{N}_{ik}(\mathbf{x}_{i,k})$. Specifically, it considers an increasing sequence of $d = 1, \ldots, 4$, and iteratively builds the sets

$$\widehat{\mathcal{N}}^d(\mathbf{x}_{i,k}) = \{\boldsymbol{\omega} \in \mathcal{K} : D(\mathbf{x}_{i,k}, \boldsymbol{\omega}) = d_{ik}, n(\boldsymbol{\omega}) > \rho\},$$

until $\widehat{\mathcal{N}}^d(\mathbf{x}_{i,k}) \neq \emptyset$ or $d = 4$. We then let

$$\mathcal{N}_{ik}(\mathbf{x}_{i,k}) = \{\mathbf{x}_{i,k}\} \cup \widehat{\mathcal{N}}^d(\mathbf{x}_{i,k}).$$

For subsequent neighborhoods $\mathcal{N}_{it}(\mathbf{x}_{i,t})$, $t > k$, we use recursion

$$\mathcal{N}_{it}(\mathbf{x}_{i,t}) = \left\{\boldsymbol{\omega} : \boldsymbol{\omega} \in \mathcal{N}^{d_{it}-1}(\mathbf{x}_{i,t}) \text{ or } \boldsymbol{\omega} \in \mathcal{N}^{d_{it}}(\mathbf{x}_{i,t}) \text{ with } \boldsymbol{\omega}_{[k]} \neq \mathbf{x}_{i,t[k]}\right\},$$

where

$$\mathcal{N}^d(\boldsymbol{\nu}) = \{\boldsymbol{\omega} : \boldsymbol{\omega} \in \mathcal{K}, D(\boldsymbol{\omega}, \boldsymbol{\nu}) \leq d\}$$

is the d-neighborhood of $k$mers within Hamming distance d of $\boldsymbol{\nu}$.

A.3 EM derivation

In this section, we derive an expectation-maximization (EM) algorithm that iteratively maximizes the penalized log-likelihood (2.6).

A.3.1 E-step

For the E-step, the conditional expectation of the complete-data log-likelihood is

$$
\begin{aligned}
&E\left[\ell_c(\boldsymbol{\theta} \mid \mathcal{R}, \mathcal{S}) \mid \mathcal{R}, \boldsymbol{\theta}^*\right] \\
&= \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{R}} \Bigg\{ \sum_{\boldsymbol{\omega} \in \mathcal{N}_{ik}(\mathbf{x}_{i,k})} \left[\log \mu(\widetilde{\boldsymbol{\omega}}) + \log f_k(\mathbf{x}_{i,k}, \mathbf{y}_{i,k} \mid \boldsymbol{\omega})\right] \zeta_{i,k}(\boldsymbol{\omega}) \\
&\quad + \sum_{t=k+1}^{l} \Bigg(\sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} \log f_t(\mathbf{x}_{i[t]}, \mathbf{y}_{i[t]} \mid \boldsymbol{\omega}, \mathbf{x}_{i,t}^{-}) \zeta_{i,t}(\boldsymbol{\omega}) \\
&\quad + \sum_{(\boldsymbol{\omega}, \boldsymbol{\nu}) \in \mathcal{N}_{it}^{\otimes}(\mathbf{x}_i)} \Big[\log p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) \mathbb{1}\{\mathcal{L}(\boldsymbol{\omega}) = 1\} \\
&\qquad + \left(\log p_1(\overline{\boldsymbol{\omega}} \mid \overline{\boldsymbol{\nu}}) + \log \mu(\widetilde{\boldsymbol{\nu}}) - \log \mu(\widetilde{\boldsymbol{\omega}})\right) \mathbb{1}\{\mathcal{L}(\boldsymbol{\omega}) = 2\} \Big] \xi_{i,t}(\boldsymbol{\omega}, \boldsymbol{\nu}) \Bigg) \Bigg\},
\end{aligned}
$$

where we have used Eq. (2.3) to rewrite $p_2(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ as a function of its corresponding free parameter $p_1(\overline{\boldsymbol{\omega}} \mid \overline{\boldsymbol{\nu}})$ and computed the probabilities of hidden states for each position in each read,

$$
\begin{aligned}
\xi_{i,t}(\boldsymbol{\omega}, \boldsymbol{\nu}) &= P\left(\mathbf{s}_{i,t-1} = \boldsymbol{\omega}, \mathbf{s}_{i,t} = \boldsymbol{\nu} \mid \mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}^*\right), \\
\zeta_{i,t}(\boldsymbol{\omega}) &= P\left(\mathbf{s}_{i,t} = \boldsymbol{\omega} \mid \mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}^*\right),
\end{aligned}
$$

given the current parameter vector $\boldsymbol{\theta}^*$. We also define the total expected number of transitions $\boldsymbol{\omega}$ to $\boldsymbol{\nu}$ in the dataset as

$$
\xi(\boldsymbol{\omega}, \boldsymbol{\nu}) = \sum_{i=1}^{|\mathcal{R}|} \sum_{t=1}^{l} \xi_{i,t}(\boldsymbol{\omega}, \boldsymbol{\nu})
$$

and the expected number of transitions between these $k$mers in either direction as

$$
\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) = \xi(\boldsymbol{\omega}, \boldsymbol{\nu}) + \xi\big(\overline{\boldsymbol{\nu}}, \overline{\boldsymbol{\omega}}\big).
$$

A.3.2 M-step

It follows that the objective function to maximize over $\boldsymbol{\theta}$ in the M-step, taking into account the penalty in Eq. (2.5), is

$$
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^*) = E\left[\ell_c(\boldsymbol{\theta} \mid \mathcal{R}, \mathcal{S}) \mid \mathcal{R}, \boldsymbol{\theta}^*\right] - \rho \mathcal{J}(\boldsymbol{\theta}).
$$

A.3.2.1 Transition distribution parameters

$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$ consists of additive terms involving groups of transition probabilities, $\{p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) : \boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})\}$, defined for each $\boldsymbol{\omega} \in \mathcal{K}$ with $\mathcal{L}(\boldsymbol{\omega}) = 1$. The M step for transition probabilities thus reduces to maximization of

$$\sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \left\{ \log p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) \widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \rho \log \left[1 + p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})/\gamma\right] \right\} - \lambda_{\boldsymbol{\omega}} \left(\sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) - 1 \right), \tag{A.1}$$

where $\lambda_{\boldsymbol{\omega}}$ is the Lagrange multiplier to impose the equality constraint in Eq. (3.1) for transitions out of $k$mer state $\boldsymbol{\omega}$. The maximum penalized-likelihood estimators (MPLEs) of the (free) transition distribution parameter are, under certain assumptions (see Theorem 1), the solution of the score functions,

$$\frac{\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu})}{p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})} - \frac{\rho}{\gamma + p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})} - \lambda_{\boldsymbol{\omega}} = 0, \tag{A.2}$$

one for each $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$. In the absence of a penalty ($\rho = 0$), the MLE is

$$\widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = \frac{\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu})}{\sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu})},$$

showing that our labeling strategy does indeed combine coverage from both strands. For positive ρ, it follows from Eq. (A.2) that $\widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ is one of

$$\frac{\phi(\boldsymbol{\omega}, \boldsymbol{\nu}) \pm \sqrt{\phi^2(\boldsymbol{\omega}, \boldsymbol{\nu}) + 4\gamma\lambda_{\boldsymbol{\omega}}\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu})}}{2\lambda_{\boldsymbol{\omega}}}, \tag{A.3}$$

where $\phi(\boldsymbol{\omega}, \boldsymbol{\nu}) = \widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \gamma\lambda_{\boldsymbol{\omega}} - \rho$ and $\lambda_{\boldsymbol{\omega}}$ is the root to the equation,

$$\sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = 1. \tag{A.4}$$

Let $\widehat{p}_1^{+}(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ and $\widehat{p}_1^{-}(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ correspond to the roots with the positive and negative sign, respectively. The following theorem indicates which root provides the solution and when the solution maximizes (A.1).

Theorem 1 *When $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) \geq \rho$ for some $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$, then there exists a unique $\lambda > 0$ satisfying Eq. (A.4), and the roots $\widehat{p}_1^{+}(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ given by Eq. (A.3) for all $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$ are real and correspond to a local maximum of Eq. (A.1).*

Proof 1 *In this proof we will partition* $\mathcal{T}(\boldsymbol{\omega}) = \mathcal{T}_{-}(\boldsymbol{\omega}) \cup \mathcal{T}_{+}(\boldsymbol{\omega})$*, where* $\boldsymbol{\nu} \in \mathcal{T}_{+}(\boldsymbol{\omega})$ *if* $\widehat{p}_1^{+}(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ *is the required solution to the score function* (A.2) *and otherwise* $\boldsymbol{\nu} \in \mathcal{T}_{-}(\boldsymbol{\omega})$*. For any* $\boldsymbol{\nu}_1 \in \mathcal{T}(\boldsymbol{\omega})$ *such that* $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}_1) \geq \rho$*, a simple analysis shows that we must have* $\lambda > 0$ *and* $\boldsymbol{\nu}_1 \in \mathcal{T}_{+}(\boldsymbol{\omega})$*, otherwise* $\widehat{p}_1(\boldsymbol{\nu}_1 \mid \boldsymbol{\omega}) < 0$*. Thus,* $\lambda > 0$ *is required. Furthermore, for any* $\boldsymbol{\nu}_2 \in \mathcal{T}(\boldsymbol{\omega})$ *with* $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}_2) < \rho$*, then* $\boldsymbol{\nu}_2 \in \mathcal{T}_{+}(\boldsymbol{\omega})$*, otherwise* $\widehat{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega}) < 0$ *because* $\lambda > 0$*. Thus* $\mathcal{T}(\boldsymbol{\omega}) = \mathcal{T}_{+}(\boldsymbol{\omega})$ *as claimed. An immediate consequence is that all* $\widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ *are real, since the discriminant is positive. WOLOG, let* $\boldsymbol{\nu}_1 = \operatorname{argmax}_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu})$*.*

We will now show that there exists $\lambda > 0$ *satisfying* (A.4) *by defining function*

$$\begin{aligned} h(\lambda) &= 2\lambda \left(\sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) - 1 \right) \\ &= -2\lambda + \sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \phi(\boldsymbol{\omega}, \boldsymbol{\nu}) \\ &\quad + \sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \sqrt{\phi^2(\boldsymbol{\omega}, \boldsymbol{\nu}) + 4\gamma\lambda\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu})}, \end{aligned} \tag{A.5}$$

when $\mathcal{T}(\boldsymbol{\omega}) = \mathcal{T}_{+}(\boldsymbol{\omega})$*. The* λ_0 *solving Eq.* (A.4) *is a zero of the function* $h(\lambda)$*. Note,*

$$h(0) = 2 \sum_{\substack{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega}) \\ \widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) > \rho}} |\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \rho| \geq 0 \tag{A.6}$$

unless $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}_1) = \rho$ *The derivative*

$$h'(0) = -2 - \gamma|\mathcal{T}(\boldsymbol{\omega})| + \gamma \sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \frac{\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) + \rho}{|\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \rho|}$$

has $\lim_{\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) \to \rho} h'(0) = \infty$ *for any* $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$ *Therefore, by continuity of* $h(\lambda)$ *for* $\lambda \geq 0$*, there exists* $\epsilon > 0$ *such that* $h(\epsilon) > 0$*. Meanwhile, we claim there exists* $L > 0$ *such that* $h(L) < 0$*, and thus by continuity of* $h(\lambda)$*, there exists a root* $\lambda_0 > 0$ *with* $h(\lambda_0) = 0$*. Specifically, we claim* $\lim_{\lambda \to \infty} h(\lambda) = -\infty$ *We rewrite* $h(\lambda)$ *to emphasize the role of* λ*.*

$$\begin{aligned} h(\lambda) &= -\lambda(2 + \gamma|\mathcal{T}(\boldsymbol{\omega})|) + \sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \left[\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \rho \right] \\ &\quad + \sum_{\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})} \lambda\gamma \left[1 + \frac{(\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \rho)^2}{\lambda^2\gamma^2} + \frac{2(\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) + \rho)}{\lambda\gamma} \right]^{1/2}. \end{aligned} \tag{A.7}$$

Apply the Taylor series expansion of $\sqrt{1+x}$ *around* $x = 0$ *to the summand*

$$\begin{aligned} & \lambda\gamma\left[1+\frac{(\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})-\rho)^2}{\lambda^2\gamma^2}+\frac{2(\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})+\rho)}{\lambda\gamma}\right]^{1/2} \\ = \;& \lambda\gamma\left[1+\frac{(\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})-\rho)^2}{2\gamma^2\lambda^2}+\frac{\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})+\rho}{\gamma\lambda}+o\left(\frac{1}{\lambda}\right)\right] \\ = \;& \lambda\gamma+o(1). \end{aligned}$$

Putting the result back in (A.8) *yields* $h(\lambda) = -2\lambda + o(1)$*, proving the claim.*

In fact, the zero, $\lambda_0 \in (0,\infty)$*, is unique because* $h''(\lambda) \leq 0$ *on* $\lambda > 0$ *The second derivative,* $h''(\lambda)$ *is*

$$\gamma^2 \sum_{\boldsymbol{\nu}\in\mathcal{T}(\boldsymbol{\omega})} \frac{\phi^2(\boldsymbol{\omega},\boldsymbol{\nu}) - \left(\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})+\gamma\lambda+\rho\right)^2 - 4\gamma\lambda\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})}{\left[\phi^2(\boldsymbol{\omega},\boldsymbol{\nu})+4\gamma\lambda\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})\right]^{3/2}}.$$

Since the denominator is positive, we need only determine the sign of the numerator, but

$$\phi^2(\boldsymbol{\omega},\boldsymbol{\nu}) - \left(\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})+\gamma\lambda+\rho\right)^2 - 4\gamma\lambda\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu}) = -4\rho\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu}),$$

which is strictly less than 0 *for all* $\lambda > 0$*.*

We now demonstrate that the solution $\widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ *using* $\lambda_0 \in (0,\infty)$ *yields a local maximum. The condition for a negative definite Hessian is*

$$\frac{\rho}{\left[\widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})+\gamma\right]^2} - \frac{\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu})}{\left[\widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})\right]^2} < 0 \tag{A.8}$$

for all $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$*. This inequality is trivially valid when* $\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu}) \geq \rho$ *for all* $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$*. Otherwise, if* $\widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu}_j) < \rho$*, write*

$$\begin{aligned} \widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu}_1) &= (1+\epsilon_1)\rho, \text{ for some } \epsilon_1 > 0, \\ \widetilde{\xi}(\boldsymbol{\omega},\boldsymbol{\nu}_j) &= (1-\epsilon_j)\rho, \text{ for some } 0 < \epsilon_j < 1, \\ \widehat{p}_1(\boldsymbol{\nu}_1 \mid \boldsymbol{\omega}) &= x, \text{ and} \\ \widehat{p}_1(\boldsymbol{\nu}_j \mid \boldsymbol{\omega}) &= \beta_j\gamma, \text{ for some } \beta_j > 0 \text{ for all } j \neq 1. \end{aligned}$$

The condition reduces to

$$\epsilon_j\beta_j^2 - 2(1-\epsilon_j)\beta_j - (1-\epsilon_j) < 0,$$

which is satisfied for β_j *between the two roots*

$$\frac{(1-\epsilon_j) \pm \left[(1-\epsilon_j)^2+\epsilon_j(1-\epsilon_j)\right]^{1/2}}{\epsilon_j}.$$

The smaller root is negative, so we require

$$\beta_j < \frac{(1-\epsilon_j) + (1-\epsilon_j)^{1/2}}{\epsilon_j}.$$

We now prove the solution of the score functions Eq. (A.2) satisfies this requirement. For all $j \neq 1$, we have

$$\frac{(1-\epsilon_j)\rho}{\beta_j\gamma} - \frac{\rho}{\gamma(1+\beta_j)} = \frac{(1+\epsilon_1)\rho}{x} - \frac{\rho}{\gamma+x}. \tag{A.9}$$

Our first claim is that $\beta_j\gamma < x$. Equation (A.9) can be rearranged to

$$\epsilon_1 x + \epsilon_j\beta_j\gamma = (\beta_j\gamma - x)\left[\frac{x\beta_j\gamma}{(\gamma+x)(\gamma+\gamma\beta_j)} - 1\right].$$

There is no solution unless $\beta_j\gamma < x$, for otherwise the left side is strictly positive and the right side is strictly negative. One can also show that $\widehat{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega}) \leq \widehat{p}_1(\boldsymbol{\nu}_1 \mid \boldsymbol{\omega})$ if $\rho \leq \widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}_2) \leq \widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}_1)$ by similar arguments. Thus, $0.25 \leq x \leq 1$.

Continuing our quest for solution β_j, we can also rearrange (A.9) into the quadratic equation

$$\frac{(\epsilon_1 x+\gamma)\gamma^2}{x^2}\beta_j^2 + \left(\frac{(\epsilon_1 x+\gamma)\gamma^2}{x^2} + \epsilon_j\gamma\right)\beta_j - (1-\epsilon_j)\gamma = 0.$$

Let $y = \frac{\epsilon_1 x+\gamma}{x^2}$, then the positive root is

$$\beta_j = \frac{-(\epsilon_j + y\gamma) + \left[(\epsilon_j+y\gamma)^2 + 4(1-\epsilon_j)y\gamma\right]^{1/2}}{2y\gamma}.$$

The bounds on x imply $\epsilon_1 < y < 2\epsilon_1 + \gamma$ is bounded above and below. Thus, the square root of the discriminant expanded about ϵ_j^2 is

$$\epsilon_j + \frac{y(1-\epsilon_j)\gamma}{\epsilon_j} + o(\gamma),$$

so

$$\begin{aligned}\beta_j &= \frac{-(\epsilon_j+y\gamma) + \epsilon_j + y\gamma\frac{2-\epsilon_j}{\epsilon_j} + o(\gamma)}{2y\gamma} \\ &= \frac{1-\epsilon_j}{\epsilon_j} + o(1) < \frac{(1-\epsilon_j)+(1-\epsilon_j)^{1/2}}{\epsilon_j}\end{aligned}$$

for sufficiently small γ, as required.

The preceding proof suggests that Newton-Raphson initialized at $\lambda = 0$ will find the root λ_0 when $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}_1) > \rho$ and there is no $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$ such that $|\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) - \rho| \leq \gamma$. When these conditions are not met, we can initialization Newton-Raphson at $\lambda = \frac{\rho}{\gamma}$, which is beyond the maximum of $h(\lambda)$ on $\lambda > 0$. The proof holds, so long as $\gamma = o(\epsilon_1)$. As $\epsilon_1 \to \gamma$, our numerical solution becomes degenerate. For example, if $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}_j) < \rho$ for all $j \neq 1$, then $\widehat{p}_1(\boldsymbol{\nu}_1 \mid \boldsymbol{\omega})$ rounds to 1. When $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) < \rho$ for all $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$, a solution to the score function exists, but it does not necessarily maximize the penalized log likelihood (details not shown). See §A.3.5 for more details about these pathological, but rare situations.

A.3.2.2 Emission distribution parameters

To update the emission parameters $g_t(\cdot)$, under the stochastic constraint $\sum_{b_x \in \Omega} g_t(b_x \mid b_s) = 1$, we maximize

$$\sum_{i=1}^{|\mathcal{R}|} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it^*}(\mathbf{x}_{i,t^*})} \log g_t(\mathbf{x}_{i[j]} \mid \boldsymbol{\omega}_{[k^*]}) \zeta_{i,t^*}(\boldsymbol{\omega}) \quad - \lambda_{b_s} \left(\sum_{b_x \in \Omega} g_t(b_x \mid b_s) - 1 \right), \tag{A.10}$$

where $t^* = \max\{t, k\}$ and $k^* = \min\{t, k\}$. The derivative with respect to $g_t(b_x \mid b_s)$ is

$$\frac{1}{g_t(b_x \mid b_s)} \zeta_t(b_x \mid b_s) - \lambda_{b_s},$$

where

$$\zeta_t(b_x \mid b_s) \quad = \quad \sum_{i=1}^{|\mathcal{R}|} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it^*}(\mathbf{x}_{i,t^*})} \zeta_{i,t^*}(\boldsymbol{\omega}) \mathbb{1}\{\boldsymbol{\omega}_{[k^*]} = b_s, \mathbf{x}_{i[t]} = b_x\}.$$

It follows that

$$\widehat{g}_t(b_x \mid b_s) = \frac{\zeta_t(b_x \mid b_s)}{\sum_{b_x \in \Omega} \zeta_t(b_x \mid b_s)}.$$

Similarly, if we define

$$\zeta_{ct}(y) = \sum_{i=1}^{|\mathcal{R}|} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it^*}(\mathbf{x}_{i,t^*})} \zeta_{i,t^*}(\boldsymbol{\omega}) \mathbb{1}\{\mathbf{y}_{i[t]} = y, \mathbf{x}_{i[t]} = \boldsymbol{\omega}_{[k^*]}\}$$

$$\zeta_{et}(y) = \sum_{i=1}^{|\mathcal{R}|} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it^*}(\mathbf{x}_{i,t^*})} \zeta_{i,t^*}(\boldsymbol{\omega}) \mathbb{1}\{\mathbf{y}_{i[t]} = y, \mathbf{x}_{i[t]} \neq \boldsymbol{\omega}_{[k^*]}\},$$

we can update q_{et} and q_{ct} as

$$\widehat{q}_{ct}(y) = \frac{\zeta_{ct}(y)}{\sum_{y'=\min_y}^{\max_y} \zeta_{ct}(y')}$$
$$\widehat{q}_{et}(y) = \frac{\zeta_{et}(y)}{\sum_{y'=\min_y}^{\max_y} \zeta_{et}(y')},$$

where $\min_y$ and $\max_y$ are the minimum and maximum quality scores.

A.3.2.3 Initial state distribution parameters

We used a Method of Moments estimator for the initial state probabilities $\mu(\widetilde{\boldsymbol{\omega}})$, *i.e.,*

$$\widehat{\mu}(\widetilde{\boldsymbol{\omega}}) \propto \widetilde{\zeta}(\widetilde{\boldsymbol{\omega}}),$$

where

$$\widetilde{\zeta}(\widetilde{\boldsymbol{\omega}}) = \sum_{i=1}^{|\mathcal{R}|} \sum_{t=k}^{l} \sum_{\boldsymbol{\omega} \in \mathcal{N}_{it}(\mathbf{x}_{i,t})} [\zeta_{i,t}(\boldsymbol{\omega}) + \zeta_{i,t}(\overline{\boldsymbol{\omega}})]$$

combines the expected number of occurrences of the canonical $k$mer $\widetilde{\boldsymbol{\omega}}$ on both strands.

A.3.3 Initialization of HMM

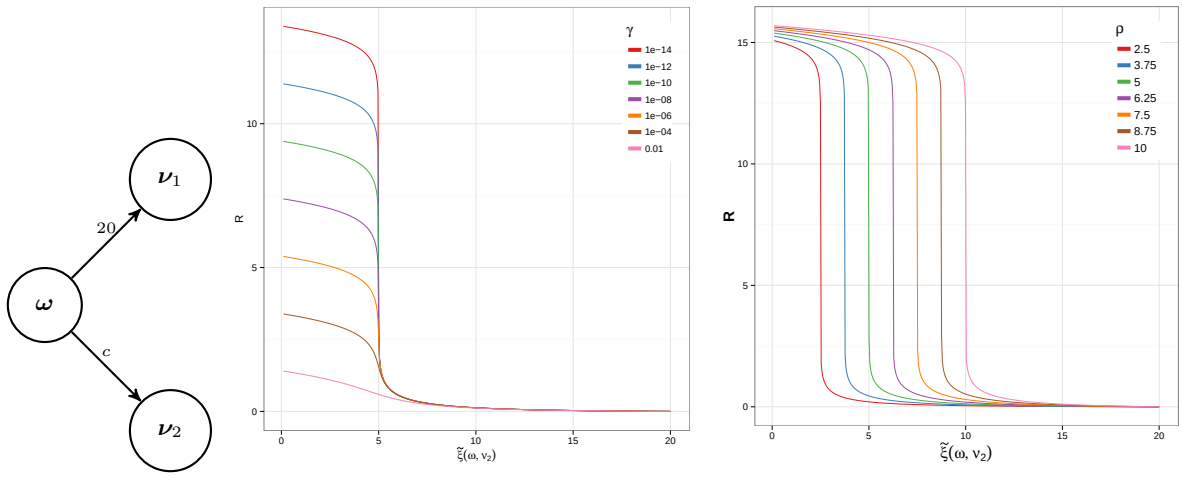
Since errors are rare, the transition parameters $p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ can be reliably initialized using the information in $\mathcal{R}$. The transition witnessed by observed $k$mers $\mathbf{x}_{i,t}$ and $\mathbf{x}_{i,t+1}$ is assigned a weight w_{it} determined by the $k+1$ quality scores,

$$w_{it} = \prod_{l=t-k+1}^{t+1} \left(1 - 10^{-\frac{\mathbf{y}_i[l]}{10}}\right).$$

Then, the weighted incidence $n_w(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ is the sum of the weights of $k$mer $\boldsymbol{\omega} \in \mathcal{K}$ followed by any $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$ observed in the reads. The combined weighted coverage,

$$\widetilde{n}_w(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = n_w(\boldsymbol{\nu} \mid \boldsymbol{\omega}) + n_w(\overline{\boldsymbol{\omega}} \mid \overline{\boldsymbol{\nu}}),$$

is the weighted number of times $\boldsymbol{\omega}$ is followed by $\boldsymbol{\nu}$ on either strand. To reduce computational complexity, we hope to remove the most obvious error transition before the EM iterations start. We sparsify transitions by applying penalty (2.5) via an M-step update using the weighted counts in place of expected counts. We account for the reduced coverage due to quality score



(a) (b) (c)

Figure A.6: The effect of the penalty function

(a) A schematic demonstrating $k$mer $\boldsymbol{\omega}$ transitioning to two possible $k$mers, $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$. The expected number of transitions is equal to 20 and c, respectively. The transition $\boldsymbol{\omega} \to \boldsymbol{\nu}_2$ is likely an error when c is close to zero. (b) As $\gamma \to 0$, the amount of shrinkage of $\widehat{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega})$ grows quickly. Regardless of the value of γ, the threshold is clearly visible at $\rho = 5$. (c) Fixing $\gamma = 10^{-16}$, the strength of the penalty is similar as the threshold, ρ, varies.

weighting by setting the penalizing threshold to $\frac{\rho}{2}$. The shrinkage factor, 0.5, for the penalizing threshold is in the neighborhood of a weighted incidence of a $k$mer with average quality score 15, for a typical $k \in [16, 30]$.

For the emission distribution, we initialize the discrete pmfs, $g_t(\cdot), q_{ct}(\cdot)$ and $q_{et}(\cdot)$ as follows.

$$g_t^{(0)}(b_1 \mid b_2) = \frac{0.01}{3} \cdot \mathbb{1}_{\{b_1 \neq b_2\}} + 0.99 \cdot \mathbb{1}_{\{b_1 = b_2\}}, \ \forall 1 \leq t \leq l, \ b_1, b_2 \in \Omega,$$
$$q_{et}^{(0)}(y) = q_{ct}(y) = \frac{1}{N_q}, \ \forall 1 \leq t \leq l, \ \min_y \leq y \leq \max_y,$$

where N_q is the number of distinct quality scores in $\mathcal{R}$.

A.3.4 The effect of the penalty function

The behavior of the penalty term and its overall impact on the estimation of transition parameters are controlled by the two parameters ρ and γ. In the text, we examined the shape of the penalty as $\gamma \to 0$ (Fig. 2.4). Here, we demonstrate how γ and ρ work jointly.

Let $\widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ denote the MPLE of $p_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ given in Eq. (A.3) and $\widetilde{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ the MLE,

$$\widetilde{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega}) = \frac{\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu})}{\sum_{\boldsymbol{\nu}' \in \mathcal{T}(\boldsymbol{\omega})} \widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}')}.$$

Now, consider a hypothetical $k$mer $\boldsymbol{\omega}$ with two allowable transitions, $\boldsymbol{\omega} \to \boldsymbol{\nu}_1$ and $\boldsymbol{\omega} \to \boldsymbol{\nu}_2$, as shown in Fig. A.5(a). Suppose $\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}_1) = 20$, and $\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}_2) = c$ for $0 < c < 20$. Without the penalty, MLE $\widetilde{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega}) = \frac{c}{c+20}$. The penalty should shrink $\widehat{p}_1(\boldsymbol{\nu}_1 \mid \boldsymbol{\omega})$ below $\widetilde{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega})$ as c declines. To measure the amount of shrinkage, define

$$R = \log_{10}\left[\frac{\widetilde{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega})}{\widehat{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega})}\right].$$

We seek parameters such that when c is small, indicative that $\boldsymbol{\omega} \to \boldsymbol{\nu}_2$ might be an error, the erroneous transition is effectively 0, say $R > 16$. On the other hand, when c is comparable to 20, R should be close to 0 to minimize the bias introduced by the penalty.

Fig. A.5(b) demonstrates the effect of γ when ρ is fixed at 5, and Fig. A.5(c) shows the effect of ρ when $\gamma = 10^{-16}$. R increases sharply when $\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}_2) \approx \rho$, so long as γ is small enough, demonstrating that the penalty behaves approximately like thresholding: $\widehat{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega})$ becomes effectively zero when $\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}_2) < \rho - \delta$, whereas $\widehat{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega}) \approx \widetilde{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega})$ when $\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}_2) > \rho + \delta$, for some small $\delta > 0$. As $\gamma \to 0$, the force of the penalty is inversely proportional to γ, so γ determines the order of magnitude of $\widehat{p}_1(\boldsymbol{\nu}_2 \mid \boldsymbol{\omega})$, when $\widetilde{\xi}^{(m)}(\boldsymbol{\omega}, \boldsymbol{\nu}_2)$ is small. For this reason, we prefer to use small γ parameters in order to achieve the maximum shrinkage of small transition probabilities. Based on our experiments, we chose $\gamma = 10^{-20}$, since smaller γ values result in numeric instability.

A.3.5 Approximations in the EM algorithm

A.3.5.1 Generalized EM

When $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) \geq \rho$ for some $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$, then we can solve for the $\widehat{p}_1(\boldsymbol{\nu} \mid \boldsymbol{\omega})$ that maximize Eq. (A.2). In the far less likely event that $\widetilde{\xi}(\boldsymbol{\omega}, \boldsymbol{\nu}) < \rho$ for all $\boldsymbol{\nu} \in \mathcal{T}(\boldsymbol{\omega})$, there is no solution to Eq. (A.2) that is a guaranteed maximum. Fortunately, the generalized EM [McLachlan and Krishnan (2007)] converges to a maximum even without requiring maximization during the M step. The only requirement is a solution in the M step that improves the log likelihood at

each iteration. We therefore solve the score function (which always has a solution, though not necessarily a maximum) and check for the log likelihood improvement at each iteration. All runs satisfied the ascent requirement of the generalized EM at every iteration.

A.3.5.2 Dropped Reads

Because we constrain the hidden state trellis to traverse through $k$mers close to the observed $k$mers as determined by d_{it} (§A.2.2) and penalty (2.5) drives some transition probabilities to numeric zero, some reads will become nearly impossible under the fitted model. Reads containing more than $\frac{k}{2}$ errors in a $k$mer become impossible when the erroneous transitions that make them possible are dropped from the model. Valid transitions may also be removed from the model, leaving some reads that traverse these lost transitions without an alternative explanation. Our solution is to drop reads that consequently become impossible. As a result, our procedure does not technically maximize the penalized log likelihood, and we risk a non-convergent algorithm since the log likelihood may always increase by dropping another read from the dataset. However, we observed that usually less than 0.1% of the change in log likelihood in early iteration is attributable to dropping reads from the likelihoods. In late iterations, this proportion ranges from 10% to 30%. In addition, both the change in log likelihood and number of dropped reads became small with increasing iterations. We conclude the algorithm is converging to a local maximum well explaining the remaining reads. We also emphasize that reads dropped from the likelihood remain in the dataset, their uncorrected errors counted against PREMIER performance.

Sufficient numeric precision can avoid the problem of impossible reads. Alternatively, one can posit a second "noise" model to generate the highly unlikely sequences, but we leave such developments to future work. After all, errors in the dropped reads typically constitute a tiny fraction of the total errors. For instance, in datasets $D1 - D3$, only 0.78%, 2.73% and 2.38% of the errors are respectively in the reads dropped by PREMIER.

A.3.6 Error correction algorithm

Once the HMM is fitted to the data, the error correction is achieved by using the Viterbi algorithm to identify the maximum likelihood "true sequence" $\mathbf{s}_i$ given the read pair $(\mathbf{x}_i, \mathbf{y}_i)$.

A.4 PREMIER software implementation

We implemented PREMIER in C/C++. PREMIER's major components, including the construction of the k-spectrum, the Baum-Welch algorithm, and the Viterbi algorithm, is parallelized using OpenMP for shared memory computers. To construct the k-spectrum, and to tally the observed transitions between $k$mers efficiently, we implemented a lock-free hash-table [Purcell and Harris (2005)]. The lock-free strategy utilizes the `compare-and-swap` (CAS) instructions to update the entries (of primitive types) in the hash-table *atomically*. The CAS operation eliminates the race conditions between concurrent threads, while incurring much smaller overheads than using a lock (*e.g.* `pthread_mutex`). Our software implementation bears strong resemblance to jellyfish [Marçais and Kingsford (2011)], but PREMIER does not require the user to specify the size of the hash-table *a priori*. To insert $k$mers into the hash-table, small batches of reads, processed in parallel, are interweaved with single-threaded sections that can resize the hash-table when necessary. The HMM can be easily parallelized at the read level because reads are assumed independent. We also exploited the CAS operation in the implementation of the Baum-Welch algorithm, where otherwise locks are needed in updating the conditional expected values.

A.5 Running the software

In this section we describe how we chose run parameters for PREMIER and other methods.

A.5.1 PREMIER run parameters

PREMIER has few run parameters. Performance is robust to most parameters chosen within a plausible range , but for best performance, the user should pay some attention to the $k$mer length, and the regularization parameter, ρ. We developed procedures for choosing

these parameters and γ, which we do not recommend adjusting, using datasets D1 and D2, and applied them without change to D3 and D4. The two other parameters, m_1 and m_2, were set once (§A.2.2) and not tuned.

The foremost parameter to determine for PREMIER, and all k-spectrum based error-correction methods, is the $k$mer length k. Choice of k should encourage $k$mer uniqueness while retaining sufficient $k$mer coverage. PREMIER requires even k to combine strand information and is limited to a maximum of $k = 30$ on 64-bit machines. For most Illumina sequencing experiments of single genomes, coverage is high enough to yield good performance even at the maximum $k = 30$. For low coverage datasets, PREMIER searches an increasing sequence of k in the range $[\widehat{k} - 4, 30]$, where $\frac{\widehat{k}}{2} = \left\lfloor \frac{\log_4(2\times 10^4 \cdot |\mathcal{G}|)+1}{2} \right\rfloor$ is a modified version of the empirical formula proposed in [Heo et al. (2014)] under the assumption of a random genome. We step through the even $k \in [\widehat{k} - 4, 30]$ and extract the k-spectrum, $\mathcal{K}^k$, from $\mathcal{R}$ and use two metrics of $k$mer uniqueness to gauge the best choice for k. Since our goal is to infer properties of the genome $\mathcal{G}$, we compute these metrics on error-filtered version of the k-spectrum. In particular, the "solid k-spectrum", $\mathcal{K}^k_{\rho_k}$, consists of $k$mers whose observed multiplicity is above a threshold ρ_k, which is chosen using the same procedure to choose the final ρ (discussed below). Then, metric R_k is the proportion of non-unique $k$mers in $\mathcal{K}^k_{\rho_k}$, *i.e.* those with more than one valid transition. Metric C_k is the model complexity of the k-th order HMM, measured by the number of state transitions in the state trellis diagram. For speed, we compute C_k from a random 1% of the high-quality reads with average quality score above 30. Clearly, both R_k, a measure of $k$mer uniqueness, and C_k, a measure of $k$mer similarity, drop as k increases and $k$mers gain distinguishable features. The k that balances coverage and uniqueness is heuristically chosen as the smallest $k \leq 28$ such that

$$\frac{R_k}{R_{k+2}} \leq 125\% \quad \text{and} \quad \frac{C_k}{C_{k+2}} \leq 125\%.$$

If there is no such k, we let $k = 30$.

For the regularization parameters ρ and γ, we note that most infrequent transitions are errors that do not actually exist in the genome. Thus, we desire an ℓ_0-like penalty, where transition probabilities are effectively set to 0 when backed by insufficient evidence. The penalty

becomes ℓ_0-like as $\gamma \to 0$ (§A.3.4), so we set $\gamma = 10^{-20}$, the smallest γ that retains numeric stability in the Newton-Raphson algorithm. The choice of the penalizing threshold, ρ, is dependent on the dataset $\mathcal{R}$. Similar to other k-spectrum based methods, after constructing the k-spectrum, we consider the histogram of $k$mer multiplicity. Most other methods use the first valley in the histogram [Liu et al. (2013); Heo et al. (2014)], but we set ρ to be half this value, plus a small constant δ, to achieve conservative performance. The added constant $\delta = 0.1$ is used to ensure that $\rho > 1$. For low-coverage datasets where the histogram is monotonically decreasing, we let $\rho = 1 + \delta$.

A.5.2 Running competing methods

To give all other competing methods an advantage, any method, except Karect [Allam et al. (2015)] because it was already competitive, that had been reported or we observed to be sensitive to run parameters was tuned on sensitive parameters and only the settings yielding the best results, as judged by gain, were reported. This section describes our tuning strategy.

A.5.2.1 BLESS

BLESS requires the user to choose the $k$mer size. Based on empirical analysis, BLESS recommends choosing the k that maximizes the number of corrected bases, while satisfying $k \geq \log_4 10^4 \cdot N_s$, where N_s is the number of distinct *solid* $k$mers in the data (a $k$mer is deemed solid if its multiplicity is above a BLESS-determined threshold). In practice, the genome length $|\mathcal{G}|$ serves as a good approximation to N_s. Therefore, to determine the optimal k for a particular dataset, we employed a linear search algorithm that starts at

$$k = \lceil \log_4(10^4 \cdot |\mathcal{G}|) \rceil \tag{A.11}$$

and increases k until the number of corrected bases reaches the maximum. If the maximum is attained at the initial value, we decrease k to find the maximum.

When the coverage of a dataset is low (*e.g.* $D2_{5\times}$ and $D3_{5\times}$), BLESS may fail to determine an appropriate threshold for the solid $k$mer multiplicity, resulting in degraded performance. In

such cases, we manually set the threshold to 2 (`-count 2`), the minimum value, in order to attain optimal performance.

To store the solid $k$mers efficiently in memory, BLESS uses a Bloom filter. Due to the stochastic nature of a Bloom filter, repeated runs of BLESS on the same dataset may yield slightly fluctuating performance. For reproducible results, we initialized the random number generator with seed 0 (via the option `-seed 0`) for all runs of BLESS in our analysis.

A.5.2.2 Coral

Since we removed all indel errors during the preparation of the datasets, we ran Coral in its "Illumina" mode (via the `-illumina` option), which suppresses the identification and correction of indel errors. The alignment scoring and penalty parameters are implied by the `-illumina` option. We kept all parameters but two at their defaults: `-k`, the $k$mer length for read indexing, and `-e`, the maximum allowed error rate in multiple sequence alignment, as Coral's performance is sensitive to both parameters.

In [Liu et al. (2013)], the authors tuned the `-e` parameter, whose default value is 0.07, on a coarse grid, $\{0.07, 0.10, 0.15, 0.20, 0.25\}$, while fixing `-k` at its default value, 21. However, we found Coral performance to also depend on the $k$mer length. Hence, we expanded the tuning process to a two-dimensional grid, spanned by $\mathbb{K}\times\mathbb{E}$, where $\mathbb{K}$ is the set of $k$mer lengths, specific to each dataset, and $\mathbb{E} = \{0.03, 0.05, 0.07, 0.10, 0.15, 0.20, 0.25\}$ is the set of error rates. The complete list of parameters we used to tune Coral is presented in Table A.2 for each dataset. The last column reports the combination of (`-k, -e`) parameters that yielded the optimal performance.

For the low coverage, subsampled datasets derived from $D1$-$D3$, we tuned the parameters on the first replicate of five and assumed the same settings for the remaining replicates. For $D4$, at each coverage level, we used the first 5 random samples (out of the 30) to tune the parameters. The optimal parameters reported in Table A.2 reflect those yielding the best average performance on the first five datasets.

Table A.2: Optimal tuning parameters for Coral

Dataset		$\mathbb{K}$	Optimal (`-k`, `-e`)
$D1$	54×	$\{11, 13, 15, 17, 19, 21, 23, 25\}$	(15, 0.10)
	10×		(13, 0.10)
	5×		(13, 0.10)
$D2$	21×	$\{19, 21, 23, 25, 27, 29, 31^*\}$	(31, 0.10)
	5×		(25, 0.07)
$D3$	30×	$\{13, 15, 17, 19, 21, 23, 25, 27\}$	(15, 0.07)
	5×		(15, 0.05)
$D4$	50×	$\{11, 13, 15, 17, 19\}$	(15, 0.20)
	40×		(15, 0.15)
	30×		(15, 0.20)
	20×		(15, 0.20)
	15×		(15, 0.15)
	10×		(13, 0.20)
	5×		(13, 0.15)

*: $k = 31$ is the largest k allowed by Coral.

A.5.2.3 HiTEC

HiTEC makes error correction decisions regarding the status of nucleotide b based on what it calls the *witness*, the upstream context of b with length w. Similar to the concept of choosing k in any k-spectrum based method, the witness length w must be determined from $\mathcal{R}$. HiTEC chooses an optimal range of w that balances uniqueness and coverage. To perform these calculations, HiTEC needs an estimate of the genome length and the approximate per-base error rate. In our experiments, we used the known genome lengths and ground truth error rates listed in Table 2.1 to run HiTEC.

A.5.2.4 Fiona

Like HiTEC, Fiona is a suffix-array based method that utilizes a range of $k$mer lengths to localize sequencing errors. The range is determined using two parameters: the estimated genome length (`-g`), and the approximate per-base error rate (`-e`). We supplied the true genome lengths found in Table 2.1 to Fiona. As for the error rate, we used the default parameter (`-e 0.01`) for all datasets. The usage of this slightly inflated error rate parameter is in accordance with

the instructions of Fiona and works well in practice, outperforming runs with error rates set to those given in Table 2.1. We ran `fiona_illumina` in the default `classifier` mode, with indel error corrections disabled (via option `-id 0`). Since Fiona performs error correction iteratively, we attempted to tune the iteration parameter (`-i`) in our test runs, but did not see improvement. Hence, this and all the other tuning parameters were left at defaults.

A.5.2.5 Karect

We used the latest version of Karect on GitHub (commit `ba3ad54`). For all datasets, we ran Karect with its default parameters, except setting `-matchtype=hamming`, considering that all indels have been removed from the datasets, and `-celltype=diploid`, provided that all datasets are derived from diploid genomes.

It is likely that Karect's performance can be further improved upon fine tuning of the alignment parameters, but there are many parameters, and we already find Karect's performance competitive or superior to PREMIER. We note that Karect exceeds PREMIER performance on D3, for which neither PREMIER nor Karect run parameters were tuned.

A.5.2.6 Musket

As a k-spectrum based method, the performance of Musket is inevitably sensitive to the selection of k. Musket does not determine k automatically, deferring the choice to the user. We ran Musket on a range of k around the value determined by the heuristic formula from BLESS Eq. (A.11), and selected the k that optimizes the error correction performance. Akin to BLESS's performance degradation on low-coverage datasets, Musket produced negative gain metrics on datasets $D2_{5\times}$ and $D3_{5\times}$. We resolved this issue by, again, manually setting the multiplicity threshold for solid $k$mers to 2 (with option `-minmulti 2`), the lowest threshold possible since Musket removes all $k$mers observed only once in $\mathcal{R}$.

Musket also has a `-maxerr` parameter that restricts the maximum number of errors allowed in any $k$mers. By default, this parameter is set to 4, and can be too conservative for correcting noisy reads. After trials and errors, we set `-maxerr=8` for all datasets, which steadily yields good performance. We also tried to increase `-maxerr` to $\lfloor \frac{k}{2} \rfloor$ to match the maximum Ham-

ming distance set for PREMIER, with mixed results: Musket's performance regresses when `-maxerr` is elevated on low-coverage datasets. Therefore, we reported all Musket results with `-maxerr=8`. Lastly, since Musket performs multiple iterations of the so-called *two-sided conservative correction*, we tried to adjust the number of itertions via the `-maxiter` option, but did not see any performance improvements. For reproducibility, the following (non-default) tuning parameters were used to generate the results for Musket:

$\mathbf{D1_{54\times}}$	$k = 25$, `-maxerr=8`
$\mathbf{D1_{10\times}}$	$k = 24$, `-maxerr=8`
$\mathbf{D1_{5\times}}$	$k = 24$, `-maxerr=8, -minmulti=2`
$\mathbf{D2_{21\times}}$	$k = 22$, `-maxerr=8`
$\mathbf{D2_{5\times}}$	$k = 23$, `-maxerr=8, -minmulti=2`
$\mathbf{D3_{5\times}}$	$k = 23$, `-maxerr=8, -minmulti=2`

A.5.2.7 Other software

In addition to the software listed in Table 2.2, we have also tested other publicly available error correction software, including k-spectrum based methods Quake [Kelley et al. (2010)], BayesHammer [Nikolenko et al. (2013)], Reptile [Yang et al. (2010)] and RACER [Ilie and Molnar (2013)], as well as suffix-tree based Shrec [Schröder et al. (2009)]. On Illumina dataset D_1, we have computed the error correction performance of these less competitive methods (Table A.3.) BayesHammer and Quake underperform in both sensitivity and gain metrics due to their aggressive trimming of erroneous reads. We exclude Shrec and Reptile from the comparison in the main text, though they both outperform Coral, due to their relative inferior performance to peer methods (HiTEC and BLESS, respectively.)

Although Reptile ranked top in an earlier survey of error correction methods [Yang et al. (2013)], there have been many improved methods since that publication. In a recent review of Illumina error-correcting methods [Molnar and Ilie (2014)], RACER was listed as one of the top-performer, along with BLESS and Musket. However, our analysis finds that despite decent sensitivity, RACER is severely undermined by an excessive number of false positives.

Table A.3: Error correction performance of BayesHammer, RACER, Quake, Shrec and Reptile on Illumina dataset D1

	BayesHammer			RACER			Quake			Shrec			Reptile		
Dataset	**sn.**	**gain**	e_p	**sn.**	**gain**	e_p	**sn.**	**gain**	e_p	**sn.**	**gain**	e_p	**sn.**	**gain**	e_p
D1 54×	0.190	0.189	0.381	0.824	0.396	0.283	0.140	0.136	0.406	0.707	0.677	0.152	0.714	0.634	0.172

A.6 Generating the benchmarking datasets

A.6.1 Datasets D1-D3: ground truth from alignment

A.6.1.1 Alignment

When a reference genome is available, sequence alignment is a popular method to identify the "ground truth" errors in $\mathcal{R}$. We used the aligner BWA [Li and Durbin (2009)] (v0.7.10) to map reads to the reference. Specifically, we used the command `bwa -a swtsw` to index the reference sequence, against which the reads were aligned using the `bwa mem` command. The default parameters of BWA-MEM algorithm "soft clip" the $5'$- or $3'$-ends when they harbor a stretch of low quality scores. For the purposes of benchmarking error-correction software, this clipping behavior is undesirable: read ends are rich in errors, but clipped errors cannot be detected. Thus, we suppressed read-clipping in BWA by specifying a large clipping penalty, `-L 100,100` when running `bwa mem`, and leaving all other alignment scoring parameters at default.

A.6.1.2 Read selection

Once reads were mapped, we discarded some reads to create a uniform dataset that all error correction algorithms could process. The output of BWA (in SAM format) was converted to BAM files using SAMtools [Li et al. (2009)]. Then, utilizing the alignment information in the BAM files, we select reads satisfying the following criteria:

- no ambiguous (`N`) bases;
- no insertion/deletion (indel) errors;
- no soft clipping (a tiny fraction of reads are soft clipped even with the `-L 100,100` option);

- no unpaired reads after the above three steps (to retain the paired-end structure).

For dataset $D1$, we also only selected reads that mapped to chromosome I of *C. elegans* genome to reduce the size of the dataset. For complete transparency, datasets $D1$ can be reproduced using the following commands:

```
$ wget ftp://ftp.wormbase.org/pub/wormbase/releases/WS238/\
   species/c_elegans/PRJNA13758/\
   c_elegans.PRJNA13758.WS238.genomic.fa.gz celeg.fa.gz
$ gunzip celeg.fa.gz
$ wget ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/\
   reads/ByRun/sra/SRR/SRR065/SRR065390/SRR065390.sra
$ fastq-dump --split-files SRR065390.sra
$ bwa index -a bwtsw celeg.fa
$ bwa mem -t 8 -L 100,100 celeg.fa SRR065390_1.fastq\
   SRR065390_2.fastq > aln.sam
$ samtools view -b -o aln.bam aln.sam
$ bam_subsample -o D1_CHR1 --reference CHROMOSOME_I \
   -I -C -N -P -E aln.bam
```

The last step utilizes a C++ program we developed (`bam_subsample`) for read selection. This program also tallies the mismatches between the selected reads and the corresponding reference genome.

Often, the mismatches reported by the aligner are directly treated as the "ground truth" errors [Yang et al. (2013); Liu et al. (2013); Schulz et al. (2014)], but such "errors" are confounded with true Single Nucleotide Polymorphisms (SNPs) between the reference and the actual sequenced genome. To remove such "non-errors", we adopted the approach in [Heo et al. (2014)], which is explained in detail next.

A.6.1.3 Generating the ground truth errors

In many situations, experimenters sequence a genome that is not identical to the "reference" genome. Because errors are rare, even slight differences between the two genomes could result

in considerable alignment mismatches and incorrectly inferred ground truth errors. To remove these variants in $D1, D2$ and $D3$, we employed `SAMtools` Li et al. (2009) to identify all single-nucleotide polymorphisms (SNPs) in the selected reads relative to the reference. The commands we used, demonstrated for $D1$, were:

```
$ samtools sort aln.bam aln-sorted
$ samtools faidx celeg.fa
$ samtools mpileup -o aln-raw.bcf --BCF \
        -f celeg.fa aln-sorted.bam
$ bcftools call -vmO b -o aln-snp.bcf aln-raw.bcf
```

Then, the list of SNPs, encoded in the BCF file, can be used to "patch" the mismatches we obtained earlier using the program `bcf_snp_patch` we developed:

```
$ awk '{ if (NR % 4 == 1) print $0; }' D1_CHR1.fastq \
   > D1_CHR1.rid
$ bcf_snp_patch -o D1_CHR1.error aln-snp.bcf aln.bam \
   D1_CHR1.rid
```

All remaining mismatches contained in the output file (`D2_CHR1.error`) were presumed to be "ground truth" sequencing errors.

A.6.2 Ground truth from duplex sequencing

The dataset $D4$ was produced by a sequencing technique called "duplex sequencing" [Schmitt et al. (2012)]. Our main motivation in using this dataset is to demonstrate the relative performance of our algorithm is not sensitive to the aligner used to generate the ground truth. These authors used duplex sequencing to detect ultra-rare mutations in the human mitochondrial genome. To detect rare mutations, their method must unambiguously distinguish mutations from the more frequent PCR and sequencing errors.

To obtain the ground truth errors, we followed the data processing procedures outlined in the supporting information of [Schmitt et al. (2012)] (cf. *Overview of Duplex Sequencing Data Processing*). For reproducibility, we note the following modifications we made to the 11-step procedure:

a) In step *(vi)*, our alignment was performed using BWA v0.7.6a (BWA-MEM algorithm with default parameters). After the alignment, we removed all reads containing "N" bases, or insertions/deletions, in addition to the reads non-mappable to the mitochondrial genome.

b) In step *(vii)* and *(viii)*, after grouping reads into "tag families" and collapsing them into SSCSs, we discarded all SSCSs containing undefined bases ("N" bases) due to low coverage or low sequence identity. For the remaining SSCSs, we created a separate data file to record the relationship between "tag families" and the SSCSs.

c) In step *(xi)*, if the two DCS partners have identical sequence, the sequence is considered as the ground truth. We then extracted all reads associated with the DCS partners (SSCSs), and tabulated all mismatches between the reads and the ground truth as sequencing errors.

A.7 Supplementary Results

A.7.1 Sensitivity to first $k$mer errors

PREMIER employs a greedy, heuristic algorithm to construct the first neighborhoods, and thus may exclude the true $k$mer, especially if the read is derived from repetitive regions. In that event, the downstream neighborhoods are affected in a cascade as well. We conducted the following experiment to measure PREMIER's sensitivity to errors presented in the first k bases of the reads.

Using the ground truth errors, we can partition the reads in each dataset into two sets: reads with *clean* first $k$mers, and reads with *erroneous* first $k$mers. After error correction, we benchmark each set to quantify the performances in the presence and absence of first $k$mer errors. The results are summarized in Table A.4.

Although reads with first $k$mer errors are generally more difficult to correct, as evident by the overall lower gains, PREMIER clearly underperforms in this group, and outperforms in the other. In particular, PREMIER generates much less false positives (measured by $\triangle = \text{sn.} - \text{gain}$) in the clean group than Karect, yet the trend is reversed in the erroneous group. The comparison here clearly indicates the underwhelming error-correction capacity for PREMIER in the first $k$mer, and the cascading effect of that: when the true $k$mer is not included in the first

Table A.4: Comparison of error correction performance for PREMIER and Karect, in reads with clean versus erroneous first $k$mers

		clean						*erroneous*					
		Karect			**PREMIER**			**Karect**			**PREMIER**		
Dataset		**sn.**	**gain**	△	**sn.**	**gain**	△	**sn.**	**gain**	△	**sn.**	**gain**	△
D1	54×	0.955	0.948	0.007	**0.959**	**0.956**	0.003	**0.852**	**0.848**	0.004	0.837	0.826	0.09
D2	21×	0.863	0.804	0.061	**0.880**	**0.845**	0.035	**0.747**	**0.732**	0.015	0.735	0.718	0.017
D3	30×	0.949	0.929	0.020	**0.941**	**0.934**	0.007	**0.929**	**0.913**	0.016	0.914	0.873	0.041

neighborhood, the HMM is forced to use alternative pathways deriving from other locations in the genome to explain the observed read, resulting in an excessive number of false positives.

A.7.2 Timing and memory information

We conducted all error correction experiments on a server with two Intel Xeon 5675 3.07GHz processors, and 96 GB of memory. All methods that support parallelization were run with eight threads.

The running time (wall clock time, denoted t) and memory (maximum resident memory, denoted **mem**) information for competing methods are summarized in Table A.5. For PREMIER, we dissected the running time, t, into three major components: $t_{\mathcal{N}}$, the time for constructing the first neighborhoods, t_{EM}, the time of a first (also the slowest) iteration of the EM algorithm, and t_{Vit}, the duration of the Viterbi algorithm. Owing to the iterative estimation of model parameters, the speed of PREMIER lags behind most competing methods. There is, however, room for optimization and improvement. Most notably, we observed that the overall running time is predominantly consumed by a small percentage of reads with large neighborhoods. Instead of computing the state trellises for these reads on the fly, caching the neighborhoods, while pruning unlikely states from them, can potentially reduce the computational overhead considerably. Also, for applications where speed is the top priority, PREMIER can be configured to directly perform error correction using the Viterbi algorithm, skipping the EM algorithm altogether. Our experiments showed that the above "fast" mode still outperforms other k, d-local methods by sizeable margin (results not shown.)

Table A.5: Timing and memory information of error correction methods in Illumina data

		BLESS		**Coral**		**Fiona**		**HiTEC**		**Karect**		**Musket**		**PREMIER**				
Dataset		t	**mem**	t	**mem**	t	**mem**	t	**mem**	t	**mem**	t	**mem**	$t_{\mathcal{N}}$	$t_{\mathbf{EM}}$	$t_{\mathbf{Vit}}$	t	**mem**
D1	54×	2m	1.4G	74m	11.2G	26m	6.6G	90m	16.2G	5m	17.8G	5m	0.2G	3m	4m	2m	27m	5.5G
	10×	23s	0.5G	12m	17.5G	4m	1.6G	13m	3.1G	6m	3.0G	50s	0.2G	2m	< 35s	11s	5m	2.5G
	5×	11s	0.4G	5m	16.7G	3m	1.2G	6m	1.6G	7m	1.5G	30s	0.2G	5m	< 15s	4s	9m	2.2G
D2	21×	9m	2.8G	94m	41.0G	80m	25.2G	414m	58.0G	40m	65.0G	18m	2.5G	75m	26m	15m	263m	37.6G
	5×	2m	1.3G	17m	29.5G	17m	5.9G	69m	16.0G	20m	14.6G	5m	1.0G	31m	4m	2m	72m	19.0G
D3	30×	19m	1.3G	307m	42.9G	146m	26.9G	430m	63.2G	37m	73.6G	27m	2.2G	11m	79m	25m	527m	34.1G
	5×	3m	1.1G	18m	23.6G	12m	3.6G	21m	10.0G	42m	8.5G	5m	0.7G	19m	8m	2m	84m	14.8G

The memory usage of PREMIER is comparable to the suffix-array based method like Fiona, and is more frugal than alignment-based methods like Coral and Karect. To further reduce the memory footprint of PREMIER, especially for high-coverage datasets, Bloom filters can be employed to filter the state space by $k$mer multiplicity, *e.g.* removing all $k$mers with unique occurrence in $\mathcal{R}$. Since a great proportion of $\mathcal{K}$ consists of erroneous $k$mers, the filtration can expectedly reduce memory consumption, and concomitantly running time, at a small trade-off for performance.

BIBLIOGRAPHY

Alexander, D. H. and Lange, K. (2011). Enhancements to the ADMIXTURE algorithm for individual ancestry estimation. BMC Bioinformatics, 12:246.

Alic, A. S., Ruzafa, D., Dopazo, J., and Blanquer, I. (2016). Objective review of de novo stand-alone error correction methods for NGS data. Wiley Interdisciplinary Reviews: Computational Molecular Science, 6(2):111–146.

Allam, A., Kalnis, P., and Solovyev, V. (2015). Karect: accurate correction of substitution, insertion and deletion errors for next-generation sequencing data. Bioinformatics, page btv415.

Armagan, A., Dunson, D. B., and Lee, J. (2013). Generalized double Pareto shrinkage. Statistica Sinica, 23(1):119–143.

Bentley, D. R., Balasubramanian, S., Swerdlow, H. P., Smith, G. P., Milton, J., Brown, C. G., Hall, K. P., Evers, D. J., Barnes, C. L., Bignell, H. R., Boutell, J. M., Bryant, J., Carter, R. J., Keira Cheetham, R., Cox, A. J., Ellis, D. J., Flatbush, M. R., Gormley, N. A., Humphray, S. J., Irving, L. J., Karbelashvili, M. S., Kirk, S. M., Li, H., Liu, X., Maisinger, K. S., Murray, L. J., Obradovic, B., Ost, T., Parkinson, M. L., Pratt, M. R., Rasolonjatovo, I. M. J., Reed, M. T., Rigatti, R., Rodighiero, C., Ross, M. T., Sabot, A., Sankar, S. V., Scally, A., Schroth, G. P., Smith, M. E., Smith, V. P., Spiridou, A., Torrance, P. E., Tzonev, S. S., Vermaas, E. H., Walter, K., Wu, X., Zhang, L., Alam, M. D., Anastasi, C., Aniebo, I. C., Bailey, D. M. D., Bancarz, I. R., Banerjee, S., Barbour, S. G., Baybayan, P. A., Benoit, V. A., Benson, K. F., Bevis, C., Black, P. J., Boodhun, A., Brennan, J. S., Bridgham, J. A., Brown, R. C., Brown, A. A., Buermann, D. H., Bundu, A. A., Burrows, J. C., Carter, N. P., Castillo, N., Chiara E. Catenazzi, M., Chang, S., Neil Cooley, R., Crake, N. R., Dada, O. O.,

Diakoumakos, K. D., Dominguez-Fernandez, B., Earnshaw, D. J., Egbujor, U. C., Elmore, D. W., Etchin, S. S., Ewan, M. R., Fedurco, M., Fraser, L. J., Fuentes Fajardo, K. V., Scott Furey, W., George, D., Gietzen, K. J., Goddard, C. P., Golda, G. S., Granieri, P. A., Green, D. E., Gustafson, D. L., Hansen, N. F., Harnish, K., Haudenschild, C. D., Heyer, N. I., Hims, M. M., Ho, J. T., Horgan, A. M., Hoschler, K., Hurwitz, S., Ivanov, D. V., Johnson, M. Q., James, T., Huw Jones, T. A., Kang, G.-D., Kerelska, T. H., Kersey, A. D., Khrebtukova, I., Kindwall, A. P., Kingsbury, Z., Kokko-Gonzales, P. I., Kumar, A., Laurent, M. A., Lawley, C. T., Lee, S. E., Lee, X., Liao, A. K., Loch, J. A., Lok, M., Luo, S., Mammen, R. M., Martin, J. W., McCauley, P. G., McNitt, P., Mehta, P., Moon, K. W., Mullens, J. W., Newington, T., Ning, Z., Ling Ng, B., Novo, S. M., O'Neill, M. J., Osborne, M. A., Osnowski, A., Ostadan, O., Paraschos, L. L., Pickering, L., Pike, A. C., Pike, A. C., Chris Pinkard, D., Pliskin, D. P., Podhasky, J., Quijano, V. J., Raczy, C., Rae, V. H., Rawlings, S. R., Chiva Rodriguez, A., Roe, P. M., Rogers, J., Rogert Bacigalupo, M. C., Romanov, N., Romieu, A., Roth, R. K., Rourke, N. J., Ruediger, S. T., Rusman, E., Sanches-Kuiper, R. M., Schenker, M. R., Seoane, J. M., Shaw, R. J., Shiver, M. K., Short, S. W., Sizto, N. L., Sluis, J. P., Smith, M. A., Ernest Sohna Sohna, J., Spence, E. J., Stevens, K., Sutton, N., Szajkowski, L., Tregidgo, C. L., Turcatti, G., vandeVondele, S., Verhovsky, Y., Virk, S. M., Wakelin, S., Walcott, G. C., Wang, J., Worsley, G. J., Yan, J., Yau, L., Zuerlein, M., Rogers, J., Mullikin, J. C., Hurles, M. E., McCooke, N. J., West, J. S., Oaks, F. L., Lundberg, P. L., Klenerman, D., Durbin, R., and Smith, A. J. (2008). Accurate whole human genome sequencing using reversible terminator chemistry. Nature, 456(7218):53–59.

Bloom, B. H. (1970). Space/Time Trade-offs in Hash Coding with Allowable Errors. Commun. ACM, 13(7):422–426.

Bragg, L. M., Stone, G., Butler, M. K., Hugenholtz, P., and Tyson, G. W. (2013). Shining a Light on Dark Sequencing: Characterising Errors in Ion Torrent PGM Data. PLoS Computational Biology, 9(4).

Bravo, H. C. and Irizarry, R. A. (2010). Model-Based Quality Assessment and Base-Calling for Second-Generation Sequencing Data. Biometrics, 66(3):665–674.

Cacho, A., Smirnova, E., Huzurbazar, S., and Cui, X. (2015). A Comparison of Base-calling Algorithms for Illumina Sequencing Technology. Briefings in Bioinformatics, page bbv088.

Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., and Fua, P. (2012). BRIEF: Computing a Local Binary Descriptor Very Fast. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(7):1281–1298.

Candès, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing Sparsity by Reweighted L1 Minimization. Journal of Fourier Analysis and Applications, 14(5-6):877–905.

Chin, C.-S., Alexander, D. H., Marks, P., Klammer, A. A., Drake, J., Heiner, C., Clum, A., Copeland, A., Huddleston, J., Eichler, E. E., Turner, S. W., and Korlach, J. (2013). Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. Nature Methods, 10(6):563–569.

Das, S. and Vikalo, H. (2013). Base calling for high-throughput short-read sequencing: dynamic programming solutions. BMC Bioinformatics, 14(1):129.

Erlich, Y., Mitra, P. P., delaBastide, M., McCombie, W. R., and Hannon, G. J. (2008). Alta-Cyclic: a self-optimizing base caller for next-generation sequencing. Nature Methods, 5(8):679–682.

Fabbro, C. D., Scalabrin, S., Morgante, M., and Giorgi, F. M. (2013). An Extensive Evaluation of Read Trimming Effects on Illumina NGS Data Analysis. PLOS ONE, 8(12):e85024.

Glenn, T. C. (2011). Field guide to next-generation DNA sequencers. Molecular Ecology Resources, 11(5):759–769.

Golan, D. and Medvedev, P. (2013). Using state machines to model the Ion Torrent sequencing process and to improve read error rates. Bioinformatics, 29(13):i344–i351.

Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q., Chen, Z., Mauceli, E., Hacohen, N., Gnirke, A., Rhind, N., di Palma, F., Birren, B. W., Nusbaum, C., Lindblad-Toh, K., Friedman, N.,

and Regev, A. (2011). Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nature Biotechnology, 29(7):644–652.

Greenfield, P., Duesing, K., Papanicolaou, A., and Bauer, D. C. (2014). Blue: correcting sequencing errors using consensus and context. Bioinformatics, 30(19):2723–2732.

Guo, Y., Ye, F., Sheng, Q., Clark, T., and Samuels, D. C. (2013). Three-stage quality control strategies for DNA re-sequencing data. Briefings in Bioinformatics, page bbt069.

Heo, Y., Wu, X.-L., Chen, D., Ma, J., and Hwu, W.-M. (2014). BLESS: Bloom filter-based error correction solution for high-throughput sequencing reads. Bioinformatics, 30(10):1354–1362.

Hoff, K. J. (2009). The effect of sequencing errors on metagenomic gene prediction. BMC Genomics, 10:520.

Huse, S. M., Welch, D. M., Morrison, H. G., and Sogin, M. L. (2010). Ironing out the wrinkles in the rare biosphere through improved OTU clustering. Environmental Microbiology, 12(7):1889–1898.

Hwang, S., Kim, E., Lee, I., and Marcotte, E. M. (2015). Systematic comparison of variant calling pipelines using gold standard personal exome variants. Scientific Reports, 5.

Ilie, L., Fazayeli, F., and Ilie, S. (2011). HiTEC: accurate error correction in high-throughput sequencing data. Bioinformatics, 27(3):295–302.

Ilie, L. and Molnar, M. (2013). RACER: Rapid and Accurate Correction of Errors in Reads. Bioinformatics, page btt407.

Johnson, D. S., Mortazavi, A., Myers, R. M., and Wold, B. (2007). Genome-Wide Mapping of in Vivo Protein-DNA Interactions. Science, 316(5830):1497–1502.

Kao, W.-C., Chan, A. H., and Song, Y. S. (2011). ECHO: A reference-free short-read error correction algorithm. Genome Research, 21(7):1181–1192.

Kao, W.-C., Stevens, K., and Song, Y. S. (2009). BayesCall: A model-based base-calling algorithm for high-throughput short-read sequencing. Genome Research, 19(10):1884–1895.

Kelley, D. R., Schatz, M. C., and Salzberg, S. L. (2010). Quake: quality-aware detection and correction of sequencing errors. Genome Biology, 11(11):R116.

Kinde, I., Wu, J., Papadopoulos, N., Kinzler, K. W., and Vogelstein, B. (2011). Detection and quantification of rare mutations with massively parallel sequencing. Proceedings of the National Academy of Sciences, 108(23):9530–9535.

Kircher, M., Stenzel, U., and Kelso, J. (2009). Improved base calling for the Illumina Genome Analyzer using machine learning strategies. Genome Biology, 10(8):1–9.

Kodama, Y., Shumway, M., and Leinonen, R. (2012). The sequence read archive: explosive growth of sequencing data. Nucleic Acids Research, 40(Database issue):D54–D56.

Kowalski, T., Grabowski, S., and Deorowicz, S. (2015). Indexing arbitrary-length k-mers in sequencing reads. arXiv:1502.01861 [cs, q-bio].

Laehnemann, D., Borkhardt, A., and McHardy, A. C. (2015). Denoising DNA deep sequencing data—high-throughput sequencing errors and their correction. Briefings in Bioinformatics, page bbv029.

Laird, P. W. (2010). Principles and challenges of genome-wide DNA methylation analysis. Nature Reviews Genetics, 11(3):191–203.

Le, H.-S., Schulz, M. H., McCauley, B. M., Hinman, V. F., and Bar-Joseph, Z. (2013). Probabilistic error correction for RNA sequencing. Nucleic Acids Research, page gkt215.

Ledergerber, C. and Dessimoz, C. (2011). Base-calling for next-generation sequencing platforms. Briefings in Bioinformatics, page bbq077.

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics, 25(14):1754–1760.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. Bioinformatics, 25(16):2078–2079.

Li, L. and Speed, T. P. (1999). An estimate of the crosstalk matrix in four-dye fluorescence-based DNA sequencing. Electrophoresis, 20(7):1433–1442.

Liu, Y., Schröder, J., and Schmidt, B. (2013). Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data. Bioinformatics, 29(3):308–315.

Loman, N. J., Misra, R. V., Dallman, T. J., Constantinidou, C., Gharbia, S. E., Wain, J., and Pallen, M. J. (2012). Performance comparison of benchtop high-throughput sequencing platforms. Nature Biotechnology, 30(5):434–439.

Lou, D. I., Hussmann, J. A., McBee, R. M., Acevedo, A., Andino, R., Press, W. H., and Sawyer, S. L. (2013). High-throughput DNA sequencing errors are reduced by orders of magnitude using circle sequencing. Proceedings of the National Academy of Sciences, 110(49):19872–19877.

Mann, T. (2015). Efficient comparison of polynucleotide sequences.

Marçais, G. and Kingsford, C. (2011). A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics, 27(6):764–770.

Mardis, E. R. (2008). The impact of next-generation sequencing technology on genetics. Trends in Genetics, 24(3):133–141.

Massingham, T. and Goldman, N. (2012). All your base: a fast and accurate probabilistic approach to base calling. Genome Biology, 13(2):1–15.

McLachlan, G. and Krishnan, T. (2007). The EM Algorithm and Extensions. John Wiley & Sons.

Medvedev, P., Scott, E., Kakaradov, B., and Pevzner, P. (2011). Error correction of high-throughput sequencing datasets with non-uniform coverage. Bioinformatics, 27(13):i137–i141.

Meng, X.-L. and Van Dyk, D. (1997). The EM Algorithm—an Old Folk-song Sung to a Fast New Tune. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 59(3):511–567.

Menges, F., Narzisi, G., and Mishra, B. (2011). TotalReCaller: improved accuracy and performance via integrated alignment and base-calling. Bioinformatics, 27(17):2330–2337.

Minoche, A. E., Dohm, J. C., and Himmelbauer, H. (2011). Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and Genome Analyzer systems. Genome Biology, 12(11):R112.

Molnar, M. and Ilie, L. (2014). Correcting Illumina data. Briefings in Bioinformatics, page bbu029.

Mullaney, J. M., Mills, R. E., Pittard, W. S., and Devine, S. E. (2010). Small insertions and deletions (INDELs) in human genomes. Human Molecular Genetics, 19(R2):R131–R136.

Nakamura, K., Oshima, T., Morimoto, T., Ikeda, S., Yoshikawa, H., Shiwa, Y., Ishikawa, S., Linak, M. C., Hirai, A., Takahashi, H., Altaf-Ul-Amin, M., Ogasawara, N., and Kanaya, S. (2011). Sequence-specific error profile of Illumina sequencers. Nucleic Acids Research, page gkr344.

Navarro, G. (2001). A Guided Tour to Approximate String Matching. ACM Comput. Surv., 33(1):31–88.

Nikolenko, S. I., Korobeynikov, A. I., and Alekseyev, M. A. (2013). BayesHammer: Bayesian clustering for error correction in single-cell sequencing. BMC Genomics, 14(1):1–11.

O'Rawe, J., Jiang, T., Sun, G., Wu, Y., Wang, W., Hu, J., Bodily, P., Tian, L., Hakonarson, H., Johnson, W. E., Wei, Z., Wang, K., and Lyon, G. J. (2013). Low concordance of multiple variant-calling pipelines: practical implications for exome and genome sequencing. Genome Medicine, 5:28.

Pevzner, P. A., Tang, H., and Waterman, M. S. (2001). An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences, 98(17):9748–9753.

Purcell, C. and Harris, T. (2005). Non-blocking Hashtables with Open Addressing. In Fraigniaud, P., editor, Distributed Computing, number 3724 in Lecture Notes in Computer Science, pages 108–121. Springer Berlin Heidelberg.

Quail, M. A., Otto, T. D., Gu, Y., Harris, S. R., Skelly, T. F., McQuillan, J. A., Swerdlow, H. P., and Oyola, S. O. (2012a). Optimal enzymes for amplifying sequencing libraries. Nature Methods, 9(1):10–11.

Quail, M. A., Smith, M., Coupland, P., Otto, T. D., Harris, S. R., Connor, T. R., Bertoni, A., Swerdlow, H. P., and Gu, Y. (2012b). A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC Genomics, 13(1):341.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286.

Renaud, G., Kircher, M., Stenzel, U., and Kelso, J. (2013). freeIbis: an efficient basecaller with calibrated quality scores for Illumina sequencers. Bioinformatics, page btt117.

Ross, M. G., Russ, C., Costello, M., Hollinger, A., Lennon, N. J., Hegarty, R., Nusbaum, C., and Jaffe, D. B. (2013). Characterizing and measuring bias in sequence data. Genome Biology, 14(5):1–20.

Rougemont, J., Amzallag, A., Iseli, C., Farinelli, L., Xenarios, I., and Naef, F. (2008). Probabilistic base calling of Solexa sequencing data. BMC Bioinformatics, 9:431.

Salmela, L. (2010). Correction of sequencing errors in a mixed set of reads. Bioinformatics, 26(10):1284–1290.

Salmela, L. and Schröder, J. (2011). Correcting errors in short reads by multiple alignments. Bioinformatics, 27(11):1455–1461.

Salzberg, S. L., Phillippy, A. M., Zimin, A., Puiu, D., Magoc, T., Koren, S., Treangen, T. J., Schatz, M. C., Delcher, A. L., Roberts, M., Marçais, G., Pop, M., and Yorke, J. A. (2012). GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome Research, 22(3):557–567.

Sanger, F., Nicklen, S., and Coulson, A. R. (1977). DNA sequencing with chain-terminating inhibitors. Proceedings of the National Academy of Sciences, 74(12):5463–5467.

Schmitt, M. W., Kennedy, S. R., Salk, J. J., Fox, E. J., Hiatt, J. B., and Loeb, L. A. (2012). Detection of ultra-rare mutations by next-generation sequencing. Proceedings of the National Academy of Sciences, 109(36):14508–14513.

Schröder, J., Schröder, H., Puglisi, S. J., Sinha, R., and Schmidt, B. (2009). SHREC: a short-read error correction method. Bioinformatics, 25(17):2157–2163.

Schulz, M. H., Weese, D., Holtgrewe, M., Dimitrova, V., Niu, S., Reinert, K., and Richard, H. (2014). Fiona: a parallel and automatic strategy for read error correction. Bioinformatics, 30(17):i356–i363.

Shokralla, S., Spall, J. L., Gibson, J. F., and Hajibabaei, M. (2012). Next-generation sequencing technologies for environmental DNA research. Molecular Ecology, 21(8):1794–1805.

Smit, A., Hubley, R., and Green, P. (1999). RepeatMasker Open-3.0. Technical report.

van Dijk, E. L., Auger, H., Jaszczyszyn, Y., and Thermes, C. (2014). Ten years of next-generation sequencing technology. Trends in Genetics, 30(9):418–426.

Wang, Z., Gerstein, M., and Snyder, M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. Nature Reviews Genetics, 10(1):57–63.

Yang, X., Chockalingam, S. P., and Aluru, S. (2013). A survey of error-correction methods for next-generation sequencing. Briefings in Bioinformatics, 14(1):56–66.

Yang, X., Dorman, K. S., and Aluru, S. (2010). Reptile: representative tiling for short read error correction. Bioinformatics, 26(20):2526–2533.